Linear Codes for Secret Sharing

Exploring Two Approaches to Generate a Secret Sharing Scheme for a Given Access Structure

Master Thesis

Raphael Markus Jean-Maxim Fehr

University of Bern

2025

Abstract

Secret sharing schemes provide a cryptographic foundation for distributing sensitive information among multiple parties such that only authorized coalitions can reconstruct the secret. This thesis studies the construction of *linear secret sharing schemes* (LSSS) for general monotone access structures, comparing two complementary approaches grounded in algebraic coding theory and threshold access trees.

The first approach, termed the *optimal approach*, exploits the correspondence between access structures and minimal codewords of dual linear codes. Following the framework of Tang, Gao, and Zhang, [9] we encode the access and adversary structures into constraint matrices G and H and characterize ideal LSSS through the orthogonality condition $GH^{\mathsf{T}} = 0$. When a solution exists, this approach yields schemes with minimal share size.

The second approach, termed the *threshold-based approach*, constructs LSSS matrices directly from threshold access trees using the algorithm of Liu, Cao, and Wong [14]. This method handles (t, n)-threshold gates natively without expanding them into Boolean formulas, achieving linear-time complexity in the number of tree leaves.

We implement both approaches in Python and evaluate them on synthetic access structures as well as on real-world quorum configurations extracted from the Stellar blockchain. Our experiments reveal a fundamental trade-off: the optimal approach guarantees minimality but becomes computationally intractable for structures with more than approximately 15 participants due to the cost of adversary structure enumeration. The threshold-based approach scales efficiently to large structures successfully constructing a 21×12 LSSS matrix for the Stellar SDF 1 validator node but does not guarantee optimal share size.

These findings provide practical guidance for LSSS construction: the optimal approach is suitable for small access structures where minimal share size is critical, while the threshold-based approach is essential for large-scale applications with hierarchical threshold policies such as blockchain consensus mechanisms.

Supervisor: Prof. Dr. Christian Cachin, Cryptology and Data Security, University of Bern
Assistant: Vivien Bammert, Cryptology and Data Security, University of Bern

Acknowledgments

I would like to express my gratitude to all those who supported me during the completion of this thesis.

My sincere thanks go to Vivien Bammert for her dedicated supervision, the helpful discussions, and the constructive feedback throughout this project.

I am equally grateful to my supervisor, Prof. Dr. Christian Cachin, for the opportunity to write this thesis in his research group, as well as for his valuable guidance and support throughout this project.

Contents

1 Introduction

Modern cryptographic systems increasingly rely on cooperation between multiple parties to protect sensitive information. Rather than entrusting a secret to a single entity, access is often governed by explicit rules specifying which groups of participants are authorised to reconstruct it. Such requirements arise in a wide range of applications, including secure key management, distributed control in organisations, and consensus mechanisms in blockchain systems [1, 15].

Secret sharing schemes provide a formal and information-theoretic framework for enforcing such access rules. In a secret sharing scheme, a trusted dealer distributes shares of a secret among a finite set of participants in such a way that only certain subsets, called *qualified sets*, can reconstruct the secret, while all other subsets learn no information about it [1, 21]. The collection of qualified sets is described by an *access structure*, which is typically assumed to be monotone: if a set of participants is authorised, then any superset is authorised as well. Formal definitions of access structures and their properties are recalled in Section 2.2.

Since their introduction by Shamir and Blakley in 1979 [7, 21] , secret sharing schemes have been studied extensively. Shamir's threshold scheme, which allows any t out of n participants to reconstruct the secret, is a canonical example of a perfect and ideal scheme. However, threshold schemes alone are often insufficient to model real-world access policies, which may involve heterogeneous roles, nested conditions, or multiple threshold requirements. To address such settings, more general constructions are needed. The first one introduced by Benaloh & Leichter [6] later formulated as *linear secret sharing schemes* (LSSS), which are introduced formally in Section 2.3.1.

A fundamental insight underlying linear secret sharing schemes is their close connection to algebraic coding theory. In particular, linear codes and their duals provide an algebraic structure that characterises which coalitions of participants can reconstruct the secret. Shamir's scheme itself admits an interpretation in terms of Reed-Solomon codes, illustrating this connection [16, 20]. More generally, properties of linear codes such as duality and minimal codewords can be used to derive and analyse secret sharing schemes for arbitrary monotone access structures. The relevant background on linear codes is reviewed in Section 2.1, while the role of minimal codewords is discussed in Section 2.1.3.

This thesis studies two complementary approaches for constructing linear secret sharing schemes for general monotone access structures. The first is a *code-based approach*, which exploits the correspondence

between access structures and minimal codewords of dual codes to obtain schemes with optimal or near-optimal share size (Chapter 4). The second is a *threshold-based approach*, which constructs LSSS matrices directly from threshold access trees and supports nested threshold policies without expanding them into large Boolean formulas (Chapter 5).

Both approaches are analysed theoretically and implemented in practice. In the practical part of the thesis, they are applied to synthetic examples as well as to access structures derived from real-world systems, including quorum configurations from the Stellar blockchain. This comparison highlights the trade offs between optimality, expressiveness, and computational efficiency in the construction of linear secret sharing schemes.

1.1 Motivation

Many modern cryptographic systems must enforce access policies that depend on cooperation between several parties rather than on the action of a single individual. Typical examples include threshold decryption in key management systems, joint control of financial resources in organisations, and quorum-based decision mechanisms in distributed consensus protocols. In all these settings, security and availability hinge on how precisely one can formalise which subsets of participants are allowed to reconstruct a secret and how to realise such policies efficiently in practice. Secret sharing schemes provide a natural and well-studied framework for this purpose. However, as systems become more complex, simple threshold schemes such as Shamir's construction are often no longer sufficient. Real-world policies may involve nested conditions, heterogeneous roles, or multi-layered organisational structures. A prominent example is the Stellar blockchain, where each validator node specifies a nested quorum set: at the top level, the node depends on a certain number of trusted organisations, and each of these organisations in turn depends on the agreement of a subset of its own validators. Such configurations are naturally described by monotone access structures with thresholds at multiple levels. This raises two central challenges. First, we need systematic methods to construct linear secret sharing schemes (LSSS) for an access structures, not just simple threshold policies. Second, these constructions must be efficient enough to be usable in practice, especially in settings where the resulting LSSS matrices are embedded into larger cryptographic systems such as secure multiparty computation or ciphertext-policy attribute-based encryption. In this thesis, we address these challenges by studying and implementing two different approaches to constructing LSSS. The first is an optimal, code-based approach that uses linear codes and minimal codewords to realise a given access structure with ideal share size whenever possible. The second is a threshold-based approach that operates directly on threshold access trees, supporting arbitrary (t,n)-gates without expanding them into large AND/OR formulas. We compare these approaches both theoretically and experimentally, and we evaluate their behaviour on synthetic examples as well as on access structures derived from Stellar quorum sets.

1.2 Background

Linear codes were originally introduced in coding theory to protect information against transmission errors by adding structured redundancy. Their algebraic properties allow efficient encoding and decoding procedures and form the foundation of modern error-correcting codes [20].

Beyond error correction, the linear structure of codes makes them suitable for cryptographic applications. In particular, linear operations over finite fields naturally support the secure distribution of information among multiple parties. This insight led to the development of *linear secret sharing schemes* (LSSS), where share generation and reconstruction are defined by linear maps[4].

In an LSSS, a secret is distributed among a set of participants such that only authorized subsets can reconstruct it, while unauthorized subsets obtain no information. The family of authorized subsets forms a monotone access structure. Massey established a fundamental connection between linear secret sharing schemes and linear codes by showing that minimal qualified sets correspond to minimal codewords of an associated linear code [16].

1.3 Research Questions

The original goal of this thesis was to understand how algebraic coding theory and secret sharing interact, and how linear codes can be used to construct linear secret sharing schemes. During the work, this focus was refined to a comparative study of two concrete construction paradigms and their behaviour on real world access structures derived from Stellar. The thesis is guided by the following research questions:

RQ1: *How can we construct linear secret sharing schemes for a given access structures using linear codes?* In particular, how do minimal codewords of the dual code characterize the access structure, and under which conditions does this yield an ideal LSSS?

RQ2: *How can we construct linear secret sharing schemes efficiently from threshold access trees?* What is the precise algorithmic procedure for generating LSSS matrices directly from nested (t, n)-gates?

RQ3: *How do the code-based "optimal" approach and the threshold-based approach compare in theory and in practice?* Specifically:

- How do the resulting LSSS matrices differ in size and structure for the same access structure?
- What is the computational effort required by each construction?
- How do both approaches behave on realistic access structures derived from Stellar quorum sets?

The practical part of the thesis addresses these questions by implementing both approaches in Python and applying them to synthetic and Stellar-based access structures, allowing for a direct empirical comparison.

1.4 Structure of the Thesis

The thesis is organised into a **theoretical part** and a **practical part**.

Theoretical Part After the introduction, Chapter 2 (Preliminaries) reviews the algebraic background needed in the rest of the work. We recall the basic notions of linear codes, including generator and parity-check matrices, dual codes, and minimum distance, and we introduce access structures and minimal codewords. Particular emphasis is placed on the connection between supports of codewords and minimal qualified sets.

Chapter 3 (Secret Sharing Schemes) formalises secret sharing and linear secret sharing schemes. We present Shamir's threshold scheme as a running example and then describe two general constructions of LSSS from linear codes, which serve as a bridge to the later code-based approach.

Chapter 4 (The Optimal Approach studies the construction proposed by Tang, Gao, and Zhang). We explain how an access structure induces an adversary structure, how to derive the constraint matrices G and H, and how the orthogonality condition $GH^{\mathsf{T}} = 0$ characterises ideal LSSS based on linear codes. Advantages and limitations of this optimal approach are discussed.

Chapter 5 (The Threshold-Based Approach) introduces the threshold access tree construction. We define threshold trees, show how to convert them into LSSS matrices without expanding thresholds into AND/OR

formulas, and present an algorithm that builds the matrix top–down by assigning share-generating vectors to nodes. We analyse the correctness and size of the resulting schemes and compare them conceptually to the optimal approach.

Practical Part The practical part applies both constructions to concrete access structures. We first describe the implementation environment and input formats for access structures, including representations as families of qualified sets and as threshold access trees. We then implement the optimal approach, detailing the computation of adversary structures, the construction of the matrices G and H, and the solution of the constraint system. Subsequently, we implement the threshold-based approach by translating threshold access trees into LSSS matrices according to the algorithm from Chapter 5. Both implementations are evaluated on synthetic examples and on access structures extracted from Stellar quorum sets. We compare matrix sizes, structural properties, and runtime behaviour. Finally, the thesis concludes with a discussion of the results, a summary of the main findings, and an outline of possible directions for future work, including extensions to more general classes of access structures and further applications in blockchain and distributed cryptography.

Part I

Theoretical Part

2
Preliminaries

This chapter introduces the fundamental concepts that form the basis for the construction of code-based secret sharing schemes considered later in this thesis. We proceed in five steps. First, we review algebraic coding theory, including generator matrices, parity-check matrices, dual codes, and Reed-Solomon codes (Section 2.1). Secondly, we introduce minimal codewords, which provide the crucial link between coding theory and secret sharing (Section 2.1.3). After that we formalize the notion of access structures, which specify which sets of participants are authorized to reconstruct a secret (Section 2.2). Third, we introduce the foundational concepts of secret sharing schemes, including definitions, security notions, and Shamir's threshold scheme as a canonical example (Section 2.3). Finally, we formalize linear secret sharing schemes (LSSS), their matrix representation, and their relationship to monotone span programs (Section 2.3.1).

2.1 Linear Codes

In the following, we give the fundamental notions of algebraic coding theory. Throughout this thesis, we work over a finite field $\mathbb{F}_q$ of order q, where q is a prime power. Instead of using the entire vector space $\mathbb{F}_q^n$, we restrict attention to linear subspaces of smaller dimension, which define the code.

Definition 2.1.1 (Linear Code)**.** [12, 20] A *linear code* $\mathcal{C}$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$. The parameter n is called the *length* of the code and k its *dimension*. The elements of $\mathcal{C}$ are called *codewords*. We refer to $\mathcal{C}$ as an $[n, k]$-linear code over $\mathbb{F}_q$.

We illustrate this with the following simple example.

Example 2.1.1. Consider the binary field $\mathbb{F}_2 = \{0, 1\}$ and the following linear subspace of $\mathbb{F}_2^3$:

$$\mathcal{C} = \big\{(0,0,0),\ (1,0,1),\ (0,1,1),\ (1,1,0)\big\}.$$

This subspace forms an $[3, 2]$-linear code over $\mathbb{F}_2$ since there are $2^2 = 4$ codewords.

A linear code $\mathcal{C}$ can be represented in two complementary ways: with a *generator matrix* that spans the code, or with a *parity-check matrix* that defines it by constraints. Equivalently, a message $m \in \mathbb{F}_q^k$ can be encoded by multiplying it with a generator matrix of $\mathcal{C}$, or, equivalently, by ensuring that the resulting codeword satisfies the parity check conditions of the code $\mathcal{C}$. In the following we give their definitions.

Definition 2.1.2 (Generator Matrix)**.** [1] A *generator matrix* G of an $[n, k]$-linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a $k \times n$ matrix whose rows form a basis of $\mathcal{C}$. Every codeword $c \in \mathcal{C}$ can be expressed as equivalently,

$$\mathcal{C} = \{\, uG \mid u \in \mathbb{F}_q^k \,\}.$$

In particular, every codeword $c \in \mathcal{C}$ can be written as

$$c = uG \quad \text{for some } u \in \mathbb{F}_q^k.$$

A generator matrix is called *systematic* if it has the form $G = [I_k \mid A]$, where I_k is the $k \times k$ identity matrix and A is a $k \times (n-k)$ matrix. In systematic form, the first k entries of each codeword directly represent the information symbols [1].

As a linear code $\mathcal{C}$ is a linear subspace of $\mathbb{F}_q^n$, it is a kernel of a linear map from $\mathbb{F}_q^n$ to $\mathbb{F}_q^{n-k}$. Therefore, there exists a matrix H, called *parity-check matrix* of $\mathcal{C}$, defined as follows.

Definition 2.1.3 (Parity-Check Matrix)**.** [1] Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be an $[n, k]$-linear code. A matrix $H \in \mathbb{F}_q^{(n-k)\times n}$ is a *parity-check matrix* for $\mathcal{C}$ if

$$\mathcal{C} = \{c \in \mathbb{F}_q^n : Hc^\top = 0\}.$$

The parity-check matrix provides a method to verify whether a received vector is a valid codeword: if $Hc^\top = 0$, then $c \in \mathcal{C}$; otherwise, an error must have occurred.

Relationship between Generator and Parity-Check Matrices [12, 20] The fundamental relationship between the generator matrix G of a code $\mathcal{C}$ and a parity-check matrix H for $\mathcal{C}$ is that the rows of G are orthogonal to the rows of H. This is given by the following theorem.

Theorem 2.1.1 (Orthogonality Condition)**.** Let $\mathcal{C}$ be an $[n, k]$-linear code with generator matrix $G \in \mathbb{F}_q^{k\times n}$ and parity-check matrix $H \in \mathbb{F}_q^{(n-k)\times n}$. Then it holds that

$$GH^T = 0.$$

Proof. By definition of the generator matrix of $\mathcal{C}$, we know that the rows of G generate $\mathcal{C}$. Moreover, by definition of the parity-check matrix, we know that for all $c \in \mathcal{C}$ it must hold that $c \cdot H^T = 0$. Therefore, since each row g of G is a codeword, it follows that each row g is orthogonal to every row of H. Thus it holds $GH^T = 0$. □

In general, there are many generator and parity check matrices for $\mathcal{C}$. If a generator matrix of a linear code $\mathcal{C}$ is in systematic form, we can find the corresponding parity-check matrix easily, given by the following corollary.

Corollary 2.1.2 (Systematic Form Construction)**.** [20] If G is in systematic form $G = [I_k \mid A]$, where I_k is the $k \times k$ identity matrix and $A \in \mathbb{F}_q^{k\times(n-k)}$, then a corresponding parity-check matrix is given by $H = [-A^T \mid I_{n-k}]$.

Proof. Direct computation shows:

$$GH^T = [I_k \mid A] \begin{bmatrix} -A \\ I_{n-k} \end{bmatrix} = -A + A = 0.$$

□

Reed-Solomon Codes Reed-Solomon codes [12, 20, Chapter 5] are an important family of linear codes that achieve optimal error-correction properties as will be discussed in Section 2.1.1. They will play a central role in Chapter 3, where we show that Shamir's secret sharing scheme can be defined using a Reed-Solomon code in a cryptographic context. Let us give its definition.

Definition 2.1.4 (Reed-Solomon Code)**.** Let k, n be integers such that $1 \leq k \leq n \leq q$ and let $\alpha_1, \ldots, \alpha_n$ be n pairwise distinct elements of $\mathbb{F}_q$. A *Reed-Solomon code* $\mathrm{RS}_q(n,k)$ is a $[n,k]$-linear code consisting of all vectors obtained by evaluating polynomials of degree at most $k-1$ at these points, that is,

$$\mathrm{RS}_q(n,k) = \big\{(f(\alpha_1), \ldots, f(\alpha_n)) \;\big|\; f(x) \in \mathbb{F}_q[x],\ \deg f \leq k-1\big\}.$$

A generator matrix of $\mathrm{RS}_q(n,k)$ is given by a $k \times n$ *Vandermonde matrix* [11] obtained by evaluating the monomials $1, x, \ldots, x^{k-1}$ at the chosen points, i.e.,

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \cdots & \alpha_n^{k-1} \end{pmatrix}.$$

Accordingly, a parity-check matrix can be chosen as an $(n-k) \times n$ Vandermonde matrix [11] evaluating the monomials $1, x, \ldots, x^{n-k-1}$ at the same evaluation points,

$$H = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & \cdots & \alpha_n^{n-k-1} \end{pmatrix},$$

so that $\mathrm{RS}_q(n,k) = \{\, c \in \mathbb{F}_q^n \mid Hc^{\mathsf{T}} = 0 \,\}$ and $GH^{\mathsf{T}} = 0$.

2.1.1 Errors and Distance

One of the main goals in coding theory is to detect and correct errors that may occur during data transmission. In most communication channels, transmitted data is subject to interference or noise, which can alter parts of the message. Without mechanisms to detect or correct such errors, this can lead to a complete loss or corruption of information. In the following, we will have a look at the error-detection and correction capabilities of an $[n,k]$-linear code $\mathcal{C}$.

Linear codes were originally introduced to add redundancy to some data in order to enable the detection and correction of errors that can occur during transmitting the data through an information channel. More specifically, when a codeword is transmitted over a noisy communication channel, some of its symbols may be altered due to noise or interference. By exploiting the algebraic structure of a code, it is often possible to detect such errors and, to a certain extent, even correct them. These error-detection and error-correction capabilities will later play an important role in the construction of secret sharing schemes.

One of the concept used to measure the error resilience of a code is the *Hamming Distance*.

Definition 2.1.5 (Hamming Distance [20, Section 1.4])**.** Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be two vectors in $\mathbb{F}_q^n$. The Hamming distance between x and y is defined as

$$d(x,y) = |\{\, i \in \{1, \ldots, n\} \mid x_i \neq y_i \,\}|.$$

Proposition 2.1.1 (Hamming Distance is a Metric)**.** [12] The Hamming distance $d : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{Z}_{\geq 0}$ is a metric on $\mathbb{F}_q^n$. That is, for all $x, y, z \in \mathbb{F}_q^n$:

1. **Non-negativity:** $d(x, y) \geq 0$, with equality if and only if $x = y$.
2. **Symmetry:** $d(x, y) = d(y, x)$.
3. **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$.

Proof. Properties (1) and (2) follow directly from the definition. For the triangle inequality, observe that if $x_i \neq z_i$ for some coordinate i, then either $x_i \neq y_i$ or $y_i \neq z_i$ (or both). Thus each coordinate contributing to $d(x, z)$ also contributes to at least one of $d(x, y)$ or $d(y, z)$. □

We will also give the notion of a *support* and the *Hamming weight*.

Definition 2.1.6 (Support)**.** [20]] Let $c = (c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n$ be a codeword of a linear code $\mathcal{C}$. The *support* of c, denoted by $supp(c)$, is the set of indices where c has non-zero entries, i.e.,

$$supp(c) = \{i \in \{1, \ldots, n\} \mid c_i \neq 0\}.$$

Definition 2.1.7 (Hamming Weight)**.** Let $v = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$. The Hamming weight of v is

$$wt(v) = |supp(v)|.$$

With this definition, we can define the minimum distance of a linear code $\mathcal{C}$. We then write for a linear code of length n, dimension k, and minimum distance d as a $[n, k, d]$-linear code.

Definition 2.1.8 (Minimum Distance [20])**.** The minimum distance d of a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is defined as

$$d = \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d(x, y).$$

Equivalently, since $\mathcal{C}$ is linear, d is the minimum Hamming weight of any non-zero codeword, i.e.,

$$d = \min_{c \in C \setminus \{0\}} \mathrm{wt}(c),$$

where $\mathrm{wt}(c) = |supp(c)|$ denotes the number of non zero coordinates of c.

Since $\mathcal{C}$ is linear, for all $x, y \in \mathcal{C}$ it holds $x - y \in \mathcal{C}$. Moreover, for any $v \in \mathbb{F}_q^n$ the Hamming distance to the 0 vector of length n is equal to the the Hamming weight of v, i.e., $d(v, 0) = wt(v)$. By translation invariance of Hamming distance, for all $x, y \in \mathcal{C}$ with $x \neq y$ it holds that

$$d(x, y) = d(x - y, 0) = wt(x - y).$$

Therefore we get the following corollary:

Corollary 2.1.3. [12] For a linear code $\mathcal{C}$ its minimum distance d is equal to the minimum weight of any non-zero codeword, i.e.,

$$\min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} d(x, y) = \min_{\substack{x,y \in \mathcal{C} \\ x \neq y}} wt(x - y).$$

Example 2.1.2. Consider the binary code

$$\mathcal{C} = \{(0,0,0),\ (1,0,1),\ (0,1,1),\ (1,1,0)\}.$$

defined as in Example 2.1.1. The non-zero codewords all have weight 2, so the minimum distance of $\mathcal{C}$ is $d = 2$ by Corollary 2.1.3. As we will see this means that $\mathcal{C}$ can detect a single error. Therefore, if we receive a vector $y \in \mathbb{F}_q^n$ that differs in more than two positions of a codeword c, we get another codeword, so we can not detect these errors.

Theorem 2.1.4 (Singleton Bound [20, Theorem 1.9])**.** Let $\mathcal{C}$ be an $[n, k, d]$-linear code $\mathbb{F}_q$. Then

$$d \ \leq \ n - k + 1.$$

Definition 2.1.9 (MDS Code [20, Chapter 5])**.** An $[n, k, d]$ linear code is called a *maximum-distance separable (MDS) code* if it attains the Singleton bound with equality, i.e.,

$$d = n - k + 1.$$

Example 2.1.3 (Minimum Distance of General Reed-Solomon Codes)**.** Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ be a vector of pairwise distinct elements of $\mathbb{F}_q$, and let

$$\mathrm{RS}_q(\boldsymbol{\alpha}, k) = \big\{ \big(f(\alpha_1), \ldots, f(\alpha_n)\big) \ \big|\ f \in \mathbb{F}_q[x],\ \deg f \leq k-1 \big\}$$

be the corresponding Reed-Solomon code. Then the minimum distance of $\mathrm{RS}_q(\boldsymbol{\alpha}, k)$ is

$$d = n - k + 1.$$

Indeed, consider two distinct codewords

$$c = \big(f(\alpha_1), \ldots, f(\alpha_n)\big), \qquad c' = \big(g(\alpha_1), \ldots, g(\alpha_n)\big),$$

where $f, g \in \mathbb{F}_q[x]$ satisfy $\deg f, \deg g \leq k-1$ and $f \neq g$. Then $h := f - g$ is a non-zero polynomial of degree at most $k-1$ and therefore has at most $k-1$ roots in $\mathbb{F}_q$. Hence c and c' agree in at most $k-1$ positions, which implies

$$d(c, c') \geq n - k + 1.$$

By the Singleton bound, equality holds. Thus, $\mathrm{RS}_q(\boldsymbol{\alpha}, k)$ is an MDS code.

One of the most fundamental limitations on linear codes is given by the *Singleton bound*. It relates the length n, dimension k, and minimum distance d of a code and shows that these parameters cannot be chosen independently.

As we have seen in the Example 2.1.3, Reed-Solomon codes attain the Singleton bound and therefore are MDS codes.

Error Detection and Correction Error corrections refer to the ability to not only recognize an error but also to correct it. Let us assume we have two links for two specific images on a website. One image is located at the place www.crypto.unibe.ch/picture/0000 and the other picture is located at the place www.crypto.unibe.ch/picture/1111. The direct call to this address will be routed to the pictures. But what if someone called the picture www.crypto.unibe.ch/picture/1110? We can map this call to the picture located in the 1111, as we see that the nearest address is 1111. Therefore, we know that an error have occurred. In this example, we are able to detect up to 3 errors. However, we can only correct one error. This can be seen as follows: If we receive 1010, we are not 100% sure if we have to map the number 1010

to either 1111 or 0000.

By the definition of the minimum distance of a linear code $\mathcal{C}$, any two distinct codewords differ in at least d positions. Therefore, a received vector that differs from a valid codeword in at most $d-1$ coordinates cannot be equal to another codeword. Hence $\mathcal{C}$ can detect up to $d-1$ errors, given by the following Lemma.

Lemma 2.1.5 (Error Detection Capability [20, Section 1.4])**.** Let $\mathcal{C}$ be a linear code with minimum distance d. Then $\mathcal{C}$ can detect up to $d-1$ symbol errors in any received vector.

Proof. Let us assume that a codeword $c \in \mathcal{C}$ is transmitted and that at most $d-1$ symbols are altered. Then the received vector $r \in \mathbb{F}_q^n$ satisfies

$$d(r, c) \leq d - 1.$$

We claim that r cannot be a (different) valid codeword than c. Suppose towards a contradiction that $r \in \mathcal{C}$ and $r \neq c$. Then, since both r and c are codewords, the definition of the minimum distance implies

$$d(r, c) \geq d,$$

which contradicts $d(r, c) \leq d - 1$. Hence $r \notin \mathcal{C}$ and therefore $\mathcal{C}$ can detect up to $d-1$ symbol errors. □

In the following we will see, how many errors a linear code $\mathcal{C}$ can correct.

Lemma 2.1.6 (Error Correction Capability [20, Section 1.4])**.** Let $\mathcal{C}$ be a linear code with minimum distance d. Then $\mathcal{C}$ can correct up to

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor$$

symbol errors by unique decoding.

Proof. Let $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. We show that no vector in $\mathbb{F}_q^n$ can be within Hamming distance at most t of two distinct codewords of $\mathcal{C}$. Assume for contradiction that there exist two distinct codewords $c_1, c_2 \in \mathcal{C}$ and a vector $r \in \mathbb{F}_q^n$ such that

$$d(r, c_1) \leq t \qquad \text{and} \qquad d(r, c_2) \leq t.$$

By the triangle inequality for the Hamming distance it follows,

$$d(c_1, c_2) \leq d(c_1, r) + d(r, c_2) \leq 2t.$$

Since $c_1 \neq c_2$ and d is the minimum distance of the code C, we must have

$$d(c_1, c_2) \geq d.$$

Combining the two inequalities yields $d \leq 2t$, which contradicts $2t < d$, because $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ implies $2t \leq d - 1$. Therefore, for every $r \in \mathbb{F}_q^n$ there exists at most one codeword $c \in C$ such that $d(r, c) \leq t$. Consequently, if a codeword c is transmitted and at most t symbol errors occur, the received word r uniquely determines c. Hence the code can correct up to t errors by unique decoding. □

2.1.2 Dual Codes

Given a linear code $\mathcal{C}$, one can derive another code from it, called *dual code of* $\mathcal{C}$ by considering the orthogonal complement of $\mathcal{C}$. In the following we give its definition.

Definition 2.1.10 (Dual Code [20, Chapter 2.3])**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be an $[n, k]$-linear code. The *dual code* $\mathcal{C}^\perp$ is defined as

$$\mathcal{C}^\perp = \{v \in \mathbb{F}_q^n : \langle v, c\rangle = 0 \text{ for all } c \in \mathcal{C}\},$$

where $\langle v, c\rangle = \sum_{i=1}^n v_i c_i$ denotes the standard inner product over $\mathbb{F}_q^n$.

The dual code of $\mathcal{C}$ has dimension $n - k$ and length n. Moreover, its generator matrix is given by the parity-check matrix of $\mathcal{C}$ and its parity-check matrix is given by the generator matrix of $\mathcal{C}$. In the context of secret sharing (Chapter 3), this relationship will allow us to characterize authorized sets via minimal codewords of the dual code.

Example 2.1.4 (Dual of a Reed-Solomon Code)**.** Consider the Reed-Solomon code $\mathrm{RS}_5(4, 2)$ over $\mathbb{F}_5$ with evaluation points $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3, \alpha_4 = 4$. This is a $[4, 2]$-linear code consisting of evaluations of polynomials of degree at most 1. Its generator matrix is the Vandermonde matrix:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

The dual code $\mathcal{C}^\perp$ has dimension $n - k = 4 - 2 = 2$. A parity-check matrix H for $\mathcal{C}$ (equivalently, a generator matrix for $\mathcal{C}^\perp$) is:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 4 & 1 \end{pmatrix}.$$

One can verify that $GH^\mathsf{T} = 0$ over $\mathbb{F}_5$. In fact, the dual of a Reed-Solomon code is again a Reed-Solomon code (up to coordinate scaling), illustrating the rich algebraic structure of this code family.

2.1.3 Minimal Codewords

So far, we have seen how linear codes can be described by generator matrices, parity-check matrices, and their duals. We now introduce *minimal codewords*, which form a crucial bridge between coding theory and secret sharing. In particular, Massey [16] showed that minimal codewords of the dual code correspond exactly to the minimal qualified sets of an access structure (Section 2.2). To describe minimal codewords formally, we first give the notion of the *support* of a codeword.

Definition 2.1.11 (Minimal Codeword [16])**.** Let $\mathcal{C}$ be a linear code of length n over a field $\mathbb{F}$. A codeword $c \in \mathcal{C}$ is called *minimal* if

1. the first support is 1 also, $c_0 = 1$
2. $c \neq 0$ and the leftmost non-zero coordinate of c equals 1, and
3. there is no other codeword $c' \in \mathcal{C}$ satisfying (1) such that

$$\mathit{supp}(c') \subsetneq \mathit{supp}(c).$$

Example 2.1.5. Consider the $[6, 2]$-linear code $\mathcal{C}$ over $\mathbb{F}_2$

$$c_1 = (1, 1, 1, 1, 1, 1), \quad c_2 = (1, 0, 1, 0, 0, 1), \quad c_3 = (0, 1, 0, 0, 1, 1), \quad c_4 = (1, 1, 0, 0, 0, 0).$$

We determine which codewords are minimal:

- $(1, 1, 0, 0, 0, 0)$ is minimal since no other non-zero codeword has strictly smaller support contained in it.
- $(1, 0, 1, 0, 0, 1)$ is also minimal, because it does not contain $(1, 1, 1, 1, 1, 1), (0, 1, 0, 0, 1, 1)$, or $(1, 1, 0, 0, 0, 0)$
- $(1, 1, 1, 1, 1, 1)$ is not minimal, because its support contains the support of $(1, 1, 0, 0, 0, 0)$.
- $(0, 1, 0, 0, 1, 1)$ is minimal, because no other nonzero codeword has strictly smaller support contained in it.

The computation of minimal codewords of a linear code is an NP-hard problem.[9] In particular, fundamental problems in linear coding theory such as computing the minimum distance are NP-hard [22], and the task of enumerating minimal codewords is at least as hard.

2.2 Access Structures

Having established the algebraic foundation of linear codes, we now turn to the combinatorial structures that specify authorization policies in secret sharing. An *access structure* formalizes which coalitions of participants should be able to reconstruct a secret and which should not. More specifically, secret sharing Section 2.3 is concerned with distributing a secret among a set of participants so that only specific coalitions can reconstruct it. In the following we give the fundamental definitions to introduce secret sharing schemes.
Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ denote a participant set. An access structure specifies which subsets of $\mathcal{P}$ are able to reconstruct a secret, and and which subsets not. More formally,

Definition 2.2.1 (Access Structure [1, Section 1.1])**.** An *access structure* on $\mathcal{P}$ is a pair $(\Gamma, \mathcal{F})$ such that

$$\Gamma \subseteq 2^{\mathcal{P}}, \qquad \mathcal{F} = 2^{\mathcal{P}} \setminus \Gamma,$$

where Γ is the set of *qualified* subsets and $\mathcal{F}$ the set of *forbidden* subsets of $\mathcal{P}$.

The following example illustrates an access structure:

Example 2.2.1. Consider a scenario where a decision requires the cooperation of one director and one manager. Let d_1, d_2 denote directors and m_1, m_2 denote manager. Then, the qualified and forbidden sets are:

$$\Gamma = \big\{\{d_1, m_1\}, \{d_1, m_2\}, \{d_2, m_1\}, \{d_2, m_2\}, \{d_1, m_1, m_2\}, \\ \{d_2, m_1, m_2\}, \{d_1, d_2, m_1\}, \{d_1, d_2, m_2\}, \{d_1, d_2, m_1, m_2\}\big\},$$

and

$$\mathcal{F} = \big\{\varnothing, \{d_1\}, \{d_2\}, \{m_1\}, \{m_2\}, \{m_1, m_2\}, \{d_1, d_2\}\big\}.$$

One checks easily that every set in Γ contains at least one director and at least one manager. Moreover, adding further participants to a qualified set cannot revoke its ability to reconstruct the secret.

The previous example motivates *monotonicity* of an access structure. In secret sharing, access structures are assumed to be *monotone increasing*, i.e., adding participants to a qualified set must preserve its ability to reconstruct the secret. In the following we give its definition.

Definition 2.2.2 (Monotone Increasing Collection [1, Section 1.1])**.** A collection $\mathcal{X} \subseteq 2^{\mathcal{P}}$ is called monotone increasing if for every $X \in \mathcal{X}$ and every Y with $X \subseteq Y \subseteq \mathcal{P}$, we have $Y \in \mathcal{X}$.

In other words, if a subset of participants is authorized to access the secret, then any larger group (including that subset) must also be a qualified set. We illustrate this by a small example with five participants, where we define some set $\mathcal{X}$.

Example 2.2.2 (Monotone Increasing)**.** [1, Section 1.1] Let $\mathcal{P} = \{A, B, C, D\}$ be a set of participants and let $\mathcal{X} \subseteq 2^{\mathcal{P}}$ be a collection of authorized sets where each element of $\mathcal{X}$ represents a group of participants allowed to access the secret. Suppose the collection of authorized sets is the following:

$$\mathcal{X} = \big\{\{A, B\}, \{B, C\}, \{A, B, C\}, \{A, B, C, D\}\big\}.$$

Now, we can check whether the collection $\mathcal{X}$ is monotone increasing.
We can see that the collection $\mathcal{X} = \big\{\{A, B\}, \{B, C\}, \{A, B, C\}, \{A, B, C, D\}\big\}$ is monotone increasing because for each authorized set X, any superset Y of X is also an authorized set.

The monotonicity assumption is natural: if a coalition can reconstruct the secret, then adding more participants should not revoke that ability. This assumption also enables a compact representation of access structures via their *minimal qualified sets*, i.e., the smallest coalitions capable of reconstruction. In this case, every larger qualified set is simply a superset of some minimal one.

Minimal Qualified Sets and Adversary Structures [1, Section 1.1] Within an access structure, some qualified sets are redundant because they contain smaller qualified subsets. The smallest qualified coalitions are called minimal qualified sets. They describe the "irreducible" groups that can reconstruct the secret.

Definition 2.2.3 (Minimal Qualified Set)**.** [1, Section 1.1] A qualified set $X \in \Gamma$ is called a minimal qualified set $\Gamma_{\min}$ if no proper subset of X is qualified; that is, for every $Y \subsetneq X$, we have $Y \notin \Gamma$.

We can illustrate the concept of minimal qualified sets with the following example.

Example 2.2.3 (Minimal Qualified Sets)**.** [1] We consider the access structure

$$\Gamma = \big\{\{d_1, m_1\}, \{d_1, m_2\}, \{d_2, m_1\}, \{d_2, m_2\}, \{d_1, m_1, m_2\}, \{d_2, m_1, m_2\}, \\ \{d_1, d_2, m_1\}, \{d_1, d_2, m_2\}, \{d_1, d_2, m_1, m_2\}\big\},$$

from Example 2.2.1. Now, let us check whether these sets are minimal qualified sets.

- Take the set $\{d_1, m_1\} \in \Gamma$. Consider the proper subsets $Y = \{d_1\}$ and $Y = \{m_1\}$. Since $\{d_1\} \notin \Gamma$ and $\{m_1\} \notin \Gamma$, we can conclude that $\{d_1, m_1\}$ is a minimal qualified set because no proper subset of $\{d_1, m_1\}$ is authorized to access the secret.
- Consider the set $\{d_1, m_1, m_2\} \in \Gamma$. We see directly that the subset $\{d_1, m_1\} \in \Gamma$. Therefore, $\{d_1, m_1, m_2\}$ is not minimal.

Let $P = \{1, \ldots, n\}$ be the set of participants and let $\Gamma \subseteq 2^P$ be a monotone access structure. We denote by

$$\Gamma_{\min} \;=\; \{\, S \in \Gamma : \text{no proper subset of } S \text{ is in } \Gamma \,\}$$

the family of *minimal qualified sets* of Γ. Every qualified set contains at least one set in $\Gamma_{\min}$. We also define a set $R \subseteq P$ is called *forbidden* if $R \notin \Gamma$. The family of all forbidden sets is called the adversary structure. We denote by

$$\mathcal{R}_{\max} \;=\; \{\, R \subseteq P : R \notin \Gamma \text{ and no proper superset of } R \text{ is forbidden} \,\}$$

the collection of *maximal forbidden sets*.
So far, we have always described access structures as sets. Now we will see how an access structure can be represented in more structured forms. One way is a monotone Boolean formula. which represents the access structure as combination of *and* and *or* combinations. We can formally describe MBFs as:

Definition 2.2.4 (Monotone Boolean Formula)**.** [5] A *monotone Boolean formula* (MBF) over participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ is a formula ϕ built inductively as follows:

1. Each variable P_i is an MBF.
2. If ϕ_1 and ϕ_2 are MBFs, then $(\phi_1 \wedge \phi_2)$ and $(\phi_1 \vee \phi_2)$ are MBFs.

Every monotone access structure can be represented by a monotone Boolean formula: a set $S \subseteq \mathcal{P}$ is qualified if and only if the formula evaluates to true when variables in S are set to true and all others to false. [5]

Threshold Gates A natural generalization of AND and OR gates is the *threshold gate*.

Definition 2.2.5 (Threshold Gate)**.** [5] A (t, n)-*threshold gate* over inputs $x_1, \ldots, x_n$ outputs `true` if and only if at least t of the n inputs are `true`. We denote this as $\mathrm{THR}_t(x_1, \ldots, x_n)$.

Observe that:

- AND corresponds to (n, n)-threshold: all inputs must be true.
- OR corresponds to $(1, n)$-threshold: at least one input must be true.

Access Tree While we can represent access structures as collections of sets, a more structured representation is often useful. An *access tree* encodes an access structure as a monotone Boolean formula built from AND ($\wedge$) and OR ($\vee$) gates. Each leaf represents a participant, and each internal node represents a logical operation. A coalition satisfies the tree if and only if the formula evaluates to `true` when the coalition's participants are set to `true` and all others to `false`. More generally, we can use *threshold gates*: a (t, n)-gate is satisfied when at least t of its n children are satisfied. 2.2.5.

Example 2.2.4. Suppose we have three participants P_1, P_2, P_3. The access policy that "either P_1 and P_2 together, or P_3 alone can reconstruct the secret" can be written as the following boolean formula:

$$f(P_1, P_2, P_3) = (P_1 \wedge P_2) \ \vee \ P_3.$$

The qualified sets are therefore $\{P_1, P_2\}$ and $\{P_3\}$, together with any of their supersets.

Threshold System An equivalent graphical representation which we will use more often is an *access tree*. Each internal node of the tree is either an AND gate or an OR gate, and the leaves are labelled with participants. Evaluating the tree on a subset of participants determines whether the set is qualified [5, Section 2.1]. Every monotone access structure can be represented by a monotone Boolean formula and hence by an access tree. [5] We use access trees because they are particularly convenient for the constructions studied in this thesis.

Example 2.2.5. Let us take a look at the SDF 1 Node in the Stellar blockchain. We can represent the quorum system as follows:

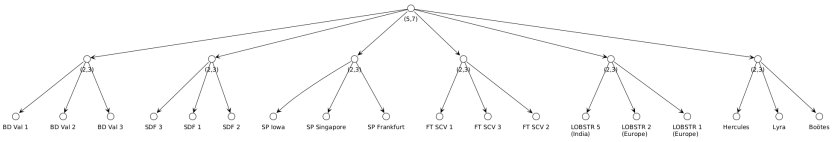

Figure 2.1: SDF 1 Access Structure

2.3 Secret Sharing Schemes

Having established the combinatorial notion of access structures, we now introduce the cryptographic primitive that realizes them: secret sharing schemes. The fundamental idea is to distribute information about a secret among multiple participants in such a way that only certain coalitions can reconstruct it, while unauthorized coalitions learn nothing.

Definition Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a set of n participants and let $\mathcal{S}$ denote the set of possible secrets. A secret sharing scheme consists of two phases: a *sharing phase* in which a dealer distributes shares to participants, and a *reconstruction phase* in which qualified coalitions recover the secret.

Definition 2.3.1 (Secret Sharing Scheme)**.** [1] A *secret sharing scheme* based on an access structure $\Gamma \subseteq 2^{\mathcal{P}}$ is a pair $(\mathsf{Share}, \mathsf{Reconstruct})$ of protocols:

- **Share protocol**: The dealer takes a secret $s \in \mathcal{S}$ and computes shares $(sh_1, \ldots, sh_n)$ from the sets $\mathcal{SH}_1, \ldots, \mathcal{SH}_n$, distributing share sh_i to each participant $p_i \in \mathcal{P}$
- **Reconstruct protocol**: A set of participants $A \subseteq \mathcal{P}$ attempts to reconstruct the secret from their shares $\{sh_i : i \in A\}$. If $A \in \Gamma$, the protocol successfully outputs the secret s

A secret sharing scheme must satisfy two fundamental requirements: The first is Correctness meaning every qualified set must be able to reconstruct the secret correctly. Formally, for any secret $s \in \mathcal{S}$ and any qualified set $A \in \Gamma$, if $(sh_1, \ldots, sh_n)$ is a valid sharing of s, then the reconstruction algorithm on input $\{sh_i : i \in A\}$ outputs s. The Second is Security meaning unauthorized coalitions should learn no information about the secret. This requirement can be formalized at different levels of strength. The strongest notion is *perfect security*:

Definition 2.3.2 (Perfect Secret Sharing)**.** [1, 3] A secret sharing scheme is perfect if for every unauthorized set $B \notin \Gamma$, the shares of participants in B reveal no information about the secret. Formally, for any two secrets $s, s' \in \mathcal{S}$ and any possible share values $\{sh_i : i \in B\}$:

$$\Pr[\{SH_i = sh_i : i \in B\} \mid S = s] = \Pr[\{SH_i = sh_i : i \in B\} \mid S = s'],$$

where the probability is taken over the randomness in the sharing algorithm.

Perfect security provides information-theoretic protection: even a computationally unbounded adversary controlling an unauthorized set learns nothing about the secret. This is in contrast to computational security, where protection relies on computational hardness assumptions.[1]

Efficiency Measures The information rate of a scheme is defined as:

Definition 2.3.3 (Information ratio)**.** [1] Let K be the finite set of secrets with $|K| \geq 2$, and let K_j denote the domain of the share given to participant p_j. The (worst-case) information ratio of a secret-sharing scheme is defined as

$$\rho \;=\; \max_{1 \leq j \leq n} \frac{\log |K_j|}{\log |K|}.$$

The quality of a secret sharing scheme is often measured by the size of shares relative to the secret:

Definition 2.3.4 (Ideal Secret Sharing)**.** [1] A secret-sharing scheme is called *ideal* if its information ratio satisfies $\rho = 1$. Equivalently, each participant receives exactly one share, and the size (information content) of each share is equal to the size of the secret.

Ideal schemes have $\rho = 1$, which is optimal. For non-ideal schemes, $\rho < 1$, meaning shares are larger than necessary.

2.3.1 Linear Secret Sharing Schemes

Having introduced the basic concepts of secret sharing, we now formalize the general notion of *linear* secret sharing schemes (LSSS). Among all secret-sharing schemes, those based on linear algebra occupy a privileged position: both the distribution of shares and the reconstruction of the secret are accomplished through linear operations over a finite field.

This linearity property offers several advantages. First, shares can be computed efficiently using matrix-vector multiplication, and reconstruction reduces to solving a linear system. Second, the security analysis leverages well-understood tools from linear algebra, making it easier to prove that unauthorized coalitions learn no information about the secret. Third, and most importantly for this thesis, linear schemes have deep connections to error-correcting codes. Indeed, the constructions we develop in Chapters 3, 4, and 5 all rely fundamentally on the linear secret sharing framework.

Definition 2.3.5 (Linear Secret Sharing Scheme)**.** [1] Let $\mathbb{F}_q$ be a finite field and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of participants. A *linear secret sharing scheme* (LSSS) over $\mathbb{F}_q$ for access structure $(\Gamma, \mathcal{F})$ is specified by a pair (M, ρ) where:

- $M \in \mathbb{F}_q^{\ell \times d}$ is a *share-generating matrix* with ℓ rows and d columns,
- $\rho : \{1, \ldots, \ell\} \to \{P_1, \ldots, P_n\}$ is a function that labels each row of M with a participant.

The scheme operates as follows:

1. **Share Distribution:** To share a secret $s \in \mathbb{F}_q$, the dealer:
   - Chooses a random vector $\mathbf{v} = (s, r_1, \ldots, r_{d-1}) \in \mathbb{F}_q^d$ where $r_1, \ldots, r_{d-1}$ are chosen uniformly at random
   - Computes the share vector $\mathbf{s} = M \cdot \mathbf{v} = (s_1, \ldots, s_\ell)^T \in \mathbb{F}_q^\ell$
   - Distributes share s_i to participant $\rho(i)$ for each $i \in \{1, \ldots, \ell\}$
2. **Secret Reconstruction:** A set of participants $A \subseteq \mathcal{P}$ is *qualified* (i.e., $A \in \Gamma$) if and only if the target vector $\mathbf{e}_1 = (1, 0, \ldots, 0)$ lies in the span of the rows corresponding to A. Formally, $A \in \Gamma$ if and only if there exist constants $\{\lambda_i : i \in I_A\}$ where $I_A = \{i : \rho(i) \in A\}$ such that:

$$\mathbf{e}_1 = \sum_{i \in I_A} \lambda_i M_i \tag{2.3.1}$$

   where M_i denotes the i-th row of M. The secret can then be recovered as:

$$s = \sum_{i \in I_A} \lambda_i s_i \tag{2.3.2}$$

Proposition 2.3.1 (Correctness of Reconstruction)**.** [1] If $\mathbf{e}_1 = \sum_{i \in I_A} \lambda_i M_i$ for some constants $\{\lambda_i : i \in I_A\}$, then the reconstruction formula (2.3.2) correctly recovers the secret, i.e., $s = \sum_{i \in I_A} \lambda_i s_i$.

Proof. Recall that the share vector is computed as $\mathbf{s} = M \cdot \mathbf{v}$, so each individual share is given by

$$s_i = M_i \cdot \mathbf{v},$$

where M_i denotes the i-th row of M (viewed as a row vector). Suppose that $\mathbf{e}_1 = \sum_{i \in I_A} \lambda_i M_i$ for some coefficients $\lambda_i \in \mathbb{F}_q$. Then:

$$\sum_{i \in I_A} \lambda_i s_i = \sum_{i \in I_A} \lambda_i (M_i \cdot \mathbf{v}) = \left(\sum_{i \in I_A} \lambda_i M_i \right) \cdot \mathbf{v} = \mathbf{e}_1 \cdot \mathbf{v}.$$

Since $\mathbf{v} = (s, r_1, \ldots, r_{d-1})$ and $\mathbf{e}_1 = (1, 0, \ldots, 0)$, we have

$$\mathbf{e}_1 \cdot \mathbf{v} = 1 \cdot s + 0 \cdot r_1 + \cdots + 0 \cdot r_{d-1} = s.$$

Therefore, $\sum_{i \in I_A} \lambda_i s_i = s$, as claimed. □

2.3.2 Shamir's Threshold Scheme

We now present Shamir's (t, n)-threshold scheme [21], which is the canonical example of an ideal and perfect secret sharing scheme. It realizes threshold access structures: any t participants can reconstruct the secret, while any $t - 1$ or fewer learn nothing about it.

The construction is based on polynomial interpolation over finite fields. The key observation is that a polynomial of degree $t - 1$ is uniquely determined by any t evaluation points, but $t - 1$ points leave one degree of freedom.

construction Shamir's Secret Sharing [21] Let q be a prime power with $q > n$, and let $\mathbb{F}_q$ be the finite field of order q. To share a secret $s \in \mathbb{F}_q$ among n participants with threshold t:

Sharing Phase

1. The dealer chooses $t - 1$ random coefficients $a_1, \ldots, a_{t-1} \in \mathbb{F}_q$ uniformly at random
2. The dealer constructs the polynomial

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \in \mathbb{F}_q[x]$$

3. The dealer chooses n distinct non-zero evaluation points $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q \setminus \{0\}$
4. Each participant p_i receives the share $sh_i = f(\alpha_i)$

Reconstruction Phase Any set of t participants can reconstruct the secret using Lagrange interpolation:

1. Given shares $\{sh_{i_1}, \ldots, sh_{i_t}\}$ from participants $\{p_{i_1}, \ldots, p_{i_t}\}$
2. Compute the Lagrange interpolation coefficients:

$$\lambda_j = \prod_{\substack{k=1 \\ k \neq j}}^{t} \frac{\alpha_{i_k}}{\alpha_{i_k} - \alpha_{i_j}} \quad \text{for } j = 1, \ldots, t$$

3. Reconstruct the secret:

$$s = f(0) = \sum_{j=1}^{t} \lambda_j \cdot sh_{i_j}$$

Theorem 2.3.1 (Correctness and Security of Shamir's Scheme)**.** [21] Shamir's (t, n)-threshold scheme is perfect and ideal:

1. **Correctness:** Any t shares uniquely determine the polynomial $f(x)$ and hence the secret $s = f(0)$
2. **Perfect Security:** Any $t - 1$ shares reveal no information about s. Specifically, for any $t - 1$ shares and any candidate secret $s^* \in \mathbb{F}_q$, there exists exactly one polynomial of degree $t - 1$ consistent with both the shares and s^*
3. **Ideality:** Each share is a single element of $\mathbb{F}_q$, same as the secret

Connection to Reed-Solomon Codes An important insight is that Shamir's scheme can be viewed as an application of Reed-Solomon codes [1, 12, 16, 21]. This establishes a fundamental connection between secret sharing and coding theory, which has been extensively studied in the literature [10, 16, 20].
Recall from Section 2.1 that a Reed-Solomon code $\mathrm{RS}_q(n, k)$ consists of evaluations of polynomials of degree at most $k - 1$ at n distinct points [19, 20]. In Shamir's (t, n)-threshold scheme:

- The secret polynomial $f(x)$ of degree $t - 1$ corresponds to encoding a message in $\mathbb{F}_q^t$ [21]
- The vector $(f(\alpha_1), \ldots, f(\alpha_n), f(0))$ is a codeword in $\mathrm{RS}_q(n + 1, t)$ [1, 16]
- The secret $s = f(0)$ occupies a designated coordinate [21]
- The shares $f(\alpha_1), \ldots, f(\alpha_n)$ are the remaining coordinates

Since Reed-Solomon codes are MDS codes [19, 20] (Section 2.1), they achieve minimum distance $d = (n+1) - t + 1 = n - t + 2$ [12, 20]. This optimal distance property means any t coordinates uniquely determine all others [16], which explains why:

- Any t shares suffice to recover s (reconstruction property) [1, 21]
- Any $t - 1$ shares provide no information about s (perfect security) [1, 21]

This observation motivates the general code-based constructions developed in Chapter 3: just as Shamir's scheme uses Reed-Solomon codes to realize threshold access structures, we can use arbitrary linear codes to realize more general monotone access structures [1, 9, 10, 16].

This concludes our review of the mathematical preliminaries.

At the end we have a linear code which allows us to detect and correct errors in a message. We can also illustrate the full concept on the following example 2.3.1:

Example 2.3.1. Consider a $[7, 4]$-RS code with minimum distance $d = n - k + 1 = 4$. Consequently, it can detect up to three errors in any codeword. Thus, if a received vector deviates from all valid codewords in at most three positions, the decoder can reliably identify that an error has occurred. If there are more errors, we will get eventually another codeword. Moreover, this code can correct up to one error.

In the following we look at an example. Assume the original messages that should be transmitted is $m = (1, 2, 3, 4) \in \mathbb{F}_q^n$. Let $m(x) = m_0 + m_1x + m_2x^2 + m_3x^3 = 1 + 2x + 3x^2 + 4x^3 \in \mathbb{F}_7[x]$

be a message polynomial, i.e., m is encoded using $f(x)$ to the codeword $c = (1,\ 3,\ 0,\ 2,\ 5,\ 5,\ 6)$ in the following way. We evaluate each position from the transmission vector.

$$
\begin{aligned}
&m(0) = 1,\\
&m(1) = 1 + 2 + 3 + 4 = 10 \equiv 3 \pmod 7,\\
&m(2) = 1 + 4 + 12 + 32 = 49 \equiv 0 \pmod 7,\\
&m(3) = 1 + 6 + 27 + 108 = 142 \equiv 2 \pmod 7,\\
&m(4) = 1 + 8 + 48 + 256 = 313 \equiv 5 \pmod 7,\\
&m(5) = 1 + 10 + 75 + 500 = 586 \equiv 5 \pmod 7,\\
&m(6) = 1 + 12 + 108 + 864 = 985 \equiv 6 \pmod 7.
\end{aligned}
$$

This lead us to the encoded message $c = (1,\ 3,\ 0,\ 2,\ 5,\ 5,\ 6)$ which we can transmit. Now, assume that the receiver gets the vector $r = (1,\ 3,\ 0,\ \mathbf{6},\ 5,\ 5,\ 6)$ is received. We can detect the error as follows: Compute the nearest nearest neighbour of the received word. If the syndrome is non-zero, then an error has occurred. Since this is a single-error correcting code with $d = 3$, the decoder checks for all possible positions to find the one where a single symbol error would produce a valid codeword. In this case, changing the fourth symbol from 6 back to 2 yields a valid codeword. Thus, we correct the error as follows: $r_{\text{corrected}} = (1,\ 3,\ 0,\ 2,\ 5,\ 5,\ 6)$, which matches the originally transmitted codeword. This is summarized in the following table.

Property	**Value**
Field	$\mathbb{F}_7$
Code parameters	$(n = 7,\ k = 4,\ d = 4)$
Error correction capability	$t = 1$ symbol
Error detection capability	up to 3 symbols
Original message	$(1,\ 2,\ 3,\ 4)$
Encoded codeword	$(1,\ 3,\ 0,\ 2,\ 5,\ 5,\ 6)$
Received word	$(1,\ 3,\ 0,\ \mathbf{6},\ 5,\ 5,\ 6)$
Corrected word	$(1,\ 3,\ 0,\ 2,\ 5,\ 5,\ 6)$

3

Code-based Linear Secret Sharing Schemes

In Chapter 2, we have established the theoretical foundations for secret sharing: Linear codes provide algebraic structure through linear and dual codes (Section 2.1), access structures specify authorization policies (Section 2.2), and Shamir's threshold scheme demonstrates how polynomial interpolation over finite fields realizes perfect and ideal secret sharing for threshold access structures (Section 2.3.2). We have also observed that Shamir's scheme admits an elegant interpretation as an application of Reed-Solomon codes, suggesting a deep connection between coding theory and secret sharing.

This chapter explores such a connection systematically. While Shamir's scheme is optimal for threshold policies, i.e., any t participants can reconstruct a secret but any $t-1$ learn nothing about the secret, real-world access structures are often more complex. Consider the organizational approval policy discussed in Chapter 1: a financial transaction might require cooperation between a director and a manager, where neither role alone suffices, but any director-manager pair is authorized. Such policies cannot be expressed as simple threshold schemes. Similarly, the nested quorum sets in the Stellar blockchain involve multi-level threshold conditions that go beyond uniform threshold access structures.

In the following we consider two approaches for the construction of secret-sharing schemes based on linear codes.

3.1 First Construction

We have seen in the Section 2.3.2 how Shamir's scheme uses polynomial evaluation to distribute shares. We now generalize this approach to arbitrary linear codes. The fundamental idea remains the same: the secret is encoded in a structured way so that only some specific coalitions can reconstruct it. However, unlike Shamir's scheme, which is limited to threshold access structures, this construction can realize any monotone access structure that can be represented by an appropriate linear code. In the following we give an overview of a first secret-sharing scheme based on linear codes.

Let $\mathcal{C}$ be an $[n,k]$-linear code over $\mathbb{F}_q$ with generator matrix

$$G = [g_1 \mid g_2 \mid \cdots \mid g_n] \in \mathbb{F}_q^{k\times n},$$

where each $g_i \in \mathbb{F}_q^k$ is a column vector of G. We have n participants $P_1, \ldots, P_n$ and a trusted dealer who holds a secret $s \in \mathbb{F}_q$. The key insight of the first construction is the following: we will embed the secret s as the *first coordinate* of a random vector $u \in \mathbb{F}_q^k$. Then, we encode u using the generator matrix of $\mathcal{C}$ to a codeword c, and then distribute the coordinates of c as shares. The access structure will then be determined entirely by the linear dependencies among the columns of G. [8]

The Sharing Protocol To share a secret $s \in \mathbb{F}_q$, a dealer proceeds as follows:

1. **Set up:** Choose a random vector

   $$u = (s, u_1, \ldots, u_{k-1}) \in \mathbb{F}_q^k$$

   where the first coordinate is the secret s and u_i for $i = 1, \ldots, k-1$ are chosen uniformly at random from $\mathbb{F}_q$. This randomness will ensure privacy against unauthorized coalitions.

2. **Encoding:** Compute

   $$y = u\,G = (u \cdot g_1,\ u \cdot g_2,\ \ldots,\ u \cdot g_n) = (s_1, s_2, \ldots, s_n).$$

   Thus, each coordinate $s_i = u \cdot g_i$ is a linear combination of the components of u, i.e.,

   $$s_i = \sum_{j=0}^{k-1} u_j G_{ji},$$

   where G_{ji} denotes the entry in row j, column i of G.

3. **Share distribution:** Participant P_i receives the share $s_i = u \cdot g_i$. In the language of Definition 2.3.5, we can view G^{T} as the share-generating matrix $M \in \mathbb{F}_q^{n \times k}$, where the i-th row of M is the column g_i^{T}. The labelling function is simply $\rho(i) = P_i$, assigning each row to its corresponding participant. The secret is embedded in the first coordinate of u, matching the convention that $\mathbf{v} = (s, r_1, \ldots, r_{k-1})$ in Definition 2.3.5.

Notice that the secret s is a single element of $\mathbb{F}_q$, and each share $s_i = u \cdot g_i$ is also a single element of $\mathbb{F}_q$. The share size equals the secret size, i.e., each participant stores exactly as much information as the secret itself, which is the best possible efficiency for a perfect secret sharing scheme.

The secret is "hidden" inside the randomness: since u has $k-1$ random components in addition to the secret, no coalition can extract s unless their shares provide enough linear constraints to isolate it.

The Reconstruction Protocol Suppose a subset of participants $I = \{i_1, i_2, \ldots, i_m\} \subseteq \{1, \ldots, n\}$ pool their shares $\{s_{i_1}, s_{i_2}, \ldots, s_{i_m}\}$. Can they recover the secret?

The answer depends on a simple linear algebraic condition. Recall that the secret s is the first component of u. We can extract this component by taking the inner product of u with the first standard basis vector $e_1 = (1, 0, \ldots, 0)^T$:

$$s = u \cdot e_1.$$

Therefore, the participants in I can reconstruct s if and only if they can express e_1 as a linear combination of their columns $\{g_{i_1}, \ldots, g_{i_m}\}$. Formally, they check whether there exist coefficients $x_1, \ldots, x_m \in \mathbb{F}_q$ such that

$$e_1 = \sum_{j=1}^{m} x_j g_{i_j}.$$

If such coefficients exist, then reconstruction is done as follows:

$$s = u \cdot e_1 = u \cdot \Big(\sum_{j=1}^{m} x_j g_{i_j}\Big) = \sum_{j=1}^{m} x_j (u \cdot g_{i_j}) = \sum_{j=1}^{m} x_j s_{i_j}.$$

Thus the participants simply compute the linear combination $\sum_j x_j s_{i_j}$ to recover s.

What if $e_1 \notin \operatorname{span}\{g_i : i \in I\}$? Then, no such coefficients exist, and the coalition cannot determine s. In fact, they learn *nothing* about s as the shares remain independent of the secret value. In the following, we will see which coalitions are authorized and that the scheme is both correct and perfectly secure.

Theorem 3.1.1 (Access Structure Characterization)**.** [1] Let $\mathcal{C}$ be an $[n, k]$-linear code over $\mathbb{F}_q$ with generator matrix

$$G = [g_1 \mid g_2 \mid \cdots \mid g_n] \in \mathbb{F}_q^{k \times n},$$

where each column vector $g_i \in \mathbb{F}_q^k$ corresponds to participant $P_i \in \mathcal{P} = \{P_1, \ldots, P_n\}$. Then, the access structure realized by the code-based secret sharing scheme described above is

$$\Gamma = \Big\{ A \subseteq \mathcal{P} \;\Big|\; e_1 \in \operatorname{span}\{g_i : P_i \in A\} \Big\}.$$

In other words, a subset of participants A is qualified if and only if the first standard basis vector $e_1 = (1, 0, \ldots, 0)^{\mathsf{T}}$ can be expressed as a linear combination of the columns of G corresponding to the participants in A.

Proof. [1, 16] We show the reconstruction and that privacy holds.

Assume that $A \subseteq \mathcal{P}$ satisfies

$$e_1 = \sum_{P_i \in A} \lambda_i g_i \quad \text{for some } \lambda_i \in \mathbb{F}_q,$$

where the column vectors g_i are exactly those columns of the generator matrix G that correspond to the participants $P_i \in A$.

Taking the inner product with $u \in \mathbb{F}_q^k$ yields

$$s = u \cdot e_1 = \sum_{P_i \in A} \lambda_i (u \cdot g_i) = \sum_{P_i \in A} \lambda_i s_i.$$

Hence, the participants in A can reconstruct the secret s as a linear combination of their shares, and therefore A is a qualified set. **(Privacy)** Conversely, suppose that

$$e_1 \notin \operatorname{span}\{g_i : P_i \in A\}.$$

By linear algebra, there exists a vector $v \in \mathbb{F}_q^k$ such that

$$v \cdot e_1 = 1 \qquad \text{and} \qquad v \cdot g_i = 0 \quad \text{for all } P_i \in A.$$

Let $s, s' \in \mathbb{F}_q$ be arbitrary secrets. For any vector u satisfying $u \cdot e_1 = s$, define

$$u' = u + (s' - s)\, v.$$

Then

$$u' \cdot e_1 = u \cdot e_1 + (s' - s)(v \cdot e_1) = s + (s' - s) = s',$$

so u' corresponds to secret s'. Moreover, for every $P_i \in A$,

$$u' \cdot g_i = u \cdot g_i + (s' - s)(v \cdot g_i) = u \cdot g_i = \sigma_i.$$

Thus the joint distribution of the shares held by A is identical for secrets s and s'. Since s and s' were arbitrary, the shares of A are statistically independent of the secret, and A is unqualified. □

Corollary 3.1.2 (Perfect Security)**.** [1] Construction 1 is a perfect secret sharing scheme, since any unauthorized coalition learns no information about the secret.

Proof. Let $I \subseteq \{1, \ldots, n\}$ be unauthorized. By Theorem 3.1.1, $e_1 \notin \operatorname{span}\{g_i : i \in I\}$. The second part of the proof of Theorem 3.1.1 shows that for any $s, s' \in F_q$, the joint distribution of the shares $(\sigma_i)_{i \in I}$ is the same whether the secret is s or s'. Equivalently, the shares of I are statistically independent of the secret, i.e., I learns no information about s. Hence the scheme is perfect [1] [9]. □

Proposition 3.1.1 (Ideal)**.** [1] Construction 1 is ideal, i.e., each share has the same size as the secret.

Proof. The secret space is $\mathbb{F}_q$, so the secret s is one field element. For each participant P_i, the share is defined as $\sigma_i = u \cdot g_i$, which is also a single element of $\mathbb{F}_q$. Thus, every share has the same size as the secret, and the scheme is ideal. □

Let us see how this works concretely with a small example over $\mathbb{F}_5$.

Example 3.1.1 (Construction 1 over $\mathbb{F}_5$)**.** Consider an $[3, 2]$-linear code $\mathcal{C}$ over $\mathbb{F}_5$. Choose a generator matrix

$$G = \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \end{pmatrix} = [g_1 \mid g_2 \mid g_3], \quad \text{of } \mathcal{C} \text{ where } \quad g_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},\ g_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix},\ g_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.$$

Sharing phase The dealer holds a secret $s \in \mathbb{F}_5$ and chooses a random element $r \in \mathbb{F}_5$. Set $u = (s, r)$. Then, the shares are:

$$\begin{aligned} s_1 &= u \cdot g_1 = s \cdot 1 + r \cdot 0 = s, \\ s_2 &= u \cdot g_2 = s \cdot 1 + r \cdot 1 = s + r \pmod 5, \\ s_3 &= u \cdot g_3 = s \cdot 2 + r \cdot 1 = 2s + r \pmod 5. \end{aligned}$$

Access structure. By Theorem 3.1.1, a subset $I \subseteq \{1, 2, 3\}$ is qualified if and only if $e_1 \in \operatorname{span}\{g_i : i \in I\}$. We examine all relevant subsets. For the singleton $\{1\}$, we have $g_1 = e_1$, hence $\{1\}$ is qualified. For $\{2\}$, since $g_2 = (1, 1)^\mathsf{T}$ and $\operatorname{span}\{g_2\} = \{\lambda(1, 1)^\mathsf{T} : \lambda \in \mathbb{F}_5\}$, it follows that $e_1 \notin \operatorname{span}\{g_2\}$, so $\{2\}$ is not qualified. Similarly, $e_1 \notin \operatorname{span}\{g_3\}$, and thus $\{3\}$ is not qualified. For the pair $\{2, 3\}$, we verify that

$$e_1 = 4g_2 + g_3 \pmod 5,$$

since

$$4\begin{pmatrix} 1 \\ 1 \end{pmatrix} + 1\begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ 5 \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} \pmod 5.$$

Hence, $\{2, 3\}$ is a qualified set. Any superset of a qualified set is again qualified. In particular, $\{1, 3\}$ and $\{1, 2\}$ are qualified because they contain $\{1\}$. However, these sets are not minimal. Therefore, the minimal qualified sets are

$$\Gamma_{\min} = \big\{\{1\}, \{2, 3\}\big\}.$$

Reconstruction examples

- **From** $\{1\}$**:** Since $e_1 = g_1$, we have $s = u \cdot e_1 = u \cdot g_1 = s_1$. Participant P_1 alone can recover the secret.
- **From** $\{2, 3\}$**:** Using $e_1 = 4g_2 + g_3$:

$$s = u \cdot e_1 = 4(u \cdot g_2) + (u \cdot g_3) = 4s_2 + s_3 \pmod 5.$$

Substituting $s_2 = s + r$ and $s_3 = 2s + r$:

$$4(s + r) + (2s + r) = 6s + 5r \equiv s \pmod 5.$$

Security verification Consider $\{2\}$ alone. The share $s_2 = s + r$ where r is uniform in $\mathbb{F}_5$. For any candidate secret s^* and any observed share value σ, there exists exactly one value of r (namely $r = \sigma - s^*$) that produces this outcome. Since r was chosen uniformly, all secrets are equally likely given the observation. This illustrates Corollary 3.1.2. **Numerical instance** Let $s = 3$, $r = 2$. Then:

$$(s_1, s_2, s_3) = (3,\ 3 + 2,\ 2 \cdot 3 + 2) = (3, 0, 3) \pmod 5.$$

Reconstruction: $\{1\}$ gives $s = s_1 = 3$. Coalition $\{2, 3\}$ gives $s = 4 \cdot 0 + 3 = 3$.

Construction 1 demonstrates how any linear code naturally induces a secret sharing scheme. The access structure is determined purely by the linear dependencies among columns of the generator matrix, i.e., a coalition is qualified if and only if its columns span the target vector e_1.

The scheme inherits the ideal property from the code structure (one share per participant, each one field element) and achieves perfect security through the randomness in the encoding vector. However, not every access structure can be realized in this way, i.e., only those that correspond to the span structure of some $[n, k]$- linear code over $\mathbb{F}_q$. Conversely, given a linear code, determining its induced access structure requires identifying all subsets of columns that span e_1. While checking whether a single subset is qualified can be done efficiently via linear algebra, the number of minimal qualified sets can be exponential in n, making explicit enumeration of the full access structure infeasible for large codes.

In the next section, we present a closely related construction that reserves one coordinate for the secret and distributes only the remaining coordinates as shares. This formulation provides a direct connection to minimal codewords of the dual code, as established by Massey [16]; see also [1, 10]. This alternative perspective will be essential for the optimal approach of Chapter 4.

3.2 Second Construction

As we have seen in construction 1, all n coordinates of a codeword are distributed as shares to the participants. We now present a variant that yields a deeper theoretical insight: by reserving one coordinate for the secret and distributing only the remaining coordinates, we obtain a direct correspondence between the access structure and the minimal codewords of the dual code of the underlying code. This connection was discovered by Massey [16] and provides the foundation for the optimal approach considered in Chapter 4. In the following we give an overview.

Let $\mathcal{C}$ be an $[n + 1, k]$-linear code over $\mathbb{F}_q$ with generator matrix

$$G = [g_0 \mid g_1 \mid g_2 \mid \cdots \mid g_n] \in \mathbb{F}_q^{k \times (n+1)},$$

where each $g_i \in \mathbb{F}_q^k$ is a column of G. The key difference from Construction 1 is that we now have $n + 1$ columns: the first column g_0 encodes the secret, while columns $g_1, \ldots, g_n$ correspond to the shares distributed to the n participants.

Sharing Phase To share a secret $s \in \mathbb{F}_q$:

1. **Set up:** Select $u = (u_0, u_1, \ldots, u_{k-1}) \in \mathbb{F}_q^k$ such that

$$s = u \cdot g_0.$$

   One natural choice is to set $g_0 = e_1 = (1, 0, \ldots, 0)^T$, in which case $u_0 = s$ and $u_1, \ldots, u_{k-1}$ are chosen uniformly at random.

2. **Encoding:** Compute

$$y = u\,G = (u \cdot g_0,\ u \cdot g_1,\ \ldots,\ u \cdot g_n) = (s, s_1, s_2, \ldots, s_n).$$

3. **Share distribution:** Participant P_i receives the share

$$s_i = u \cdot g_i, \quad i = 1, \ldots, n.$$

The difference is subtle but important: participants collectively hold only the "tail" of the codeword, while the "head" coordinate encodes the secret.

Reconstruction Phase A coalition $I \subseteq \{1, \ldots, n\}$ can reconstruct s if and only if the secret column g_0 lies in the span of their columns $\{g_i : i \in I\}$. If there exist coefficients $x_1, \ldots, x_m$ such that

$$g_0 = \sum_{j=1}^{m} x_j g_{i_j},$$

then reconstruction follows by the same logic as before:

$$s = u \cdot g_0 = u \cdot \Big(\sum_j x_j g_{i_j}\Big) = \sum_j x_j (u \cdot g_{i_j}) = \sum_j x_j s_{i_j}.$$

The Access Structure The access structure has the same form as in the first construction, but with the secret column g_0 replacing the target vector e_1. This is given by the following Theorem.

Theorem 3.2.1 (Access Structure Characterization)**.** [1] Let $\mathcal{C}$ be an $[n+1, k]$-linear code over $\mathbb{F}_q$ with generator matrix $G = [g_0 \mid g_1 \mid \cdots \mid g_n]$, and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of participants where P_i receives the share $s_i = u \cdot g_i$. The access structure realized by the code-based secret sharing scheme described above is

$$\Gamma = \Big\{ A \subseteq \mathcal{P} \ \Big|\ g_0 \in \operatorname{span}\{g_i : P_i \in A\} \Big\}.$$

In other words, a subset of participants A is qualified if and only if the secret column g_0 can be expressed as a linear combination of the columns of G corresponding to the participants in A.

The proofs of Theorems 3.1.1 and 3.2.1 are identical up to replacing the distinguished vector e_1 by the column g_0. Both arguments rely on the same linear-algebraic reconstruction and separation steps; Construction 2 is coordinate-free, while Construction 1 fixes the secret to a specific basis vector. For the sake of completeness, it is listed here again and not left as an exercise for the reader.

Proof. In Construction 2, the dealer embeds the secret in the coordinate corresponding to the column g_0 and distributes to each participant P_i the share $\sigma_i = u \cdot g_i$, where $u \in \mathbb{F}_q^k$ is chosen uniformly at random subject to the constraint $u \cdot g_0 = s$.

Let $A \subseteq \mathcal{P}$ be an arbitrary subset of participants, and let

$$I := \{\, i \in \{1, \ldots, n\} \mid P_i \in A \,\}$$

denote the corresponding index set. If $g_0 = \sum_{i \in I} \lambda_i g_i$, then

$$s = u \cdot g_0 = \sum_{i \in I} \lambda_i (u \cdot g_i) = \sum_{i \in I} \lambda_i \sigma_i,$$

so the participants in I can reconstruct the secret.

Conversely, if $g_0 \notin \operatorname{span}\{g_i : i \in I\}$, then there exists $v \in \mathbb{F}_q^k$ such that $v \cdot g_0 = 1$ and $v \cdot g_i = 0$ for all $i \in I$. For any two secrets s, s', the translation $u \mapsto u + (s' - s)v$ preserves all shares σ_i with $i \in I$ while changing the value of $u \cdot g_0$. Hence the joint distribution of the shares of I is independent of the secret, and I is unauthorized. Therefore, I is qualified if and only if $g_0 \in \operatorname{span}\{g_i : i \in I\}$. □

Example 3.2.1 (Example of Theorem 3.2.1)**.** Let $q = 5$ and consider an $[4, 2]$-linear code with generator matrix

$$G = \begin{bmatrix} g_0 \mid g_1 \mid g_2 \mid g_3 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

The dealer chooses $u = (s, r) \in \mathbb{F}_5^2$ with random r, so that

$$\sigma_1 = s, \qquad \sigma_2 = r, \qquad \sigma_3 = s + r.$$

By Theorem 3.2.1, a set A is qualified if and only if $g_0 \in \operatorname{span}\{g_i : P_i \in A\}$. Indeed, $\{P_1\}$ is qualified since $g_0 = g_1$, and $\{P_2, P_3\}$ is qualified since $g_0 = g_3 - g_2$. All other proper subsets are unqualified. Thus the minimal qualified set is

$$\Gamma_{\min}\{\{P_1\}, \{P_2, P_3\}\}.$$

Connection to Dual Codes and Minimal Codewords The power of the second construction lies in what happens when we examine the dual code $\mathcal{C}^\perp$. This connection was first established by Massey [16] and provides a fundamental link between coding theory and secret sharing; see also [1, 20].

Recall that a codeword $c \in \mathcal{C}^\perp$ satisfies $c \cdot y = 0$ for every $y \in \mathcal{C}$. This orthogonality condition provides an alternative characterization of which coalitions can reconstruct the secret. Let $y = (s, s_1, \ldots, s_n) \in \mathcal{C}$ be any codeword produced by the sharing phase, and let $c = (c_0, c_1, \ldots, c_n) \in \mathcal{C}^\perp$. The orthogonality condition gives:

$$c \cdot y = c_0 s + \sum_{i=1}^{n} c_i s_i = 0.$$

If $c_0 \neq 0$, we can solve for the secret:

$$s = -\frac{1}{c_0} \sum_{i=1}^{n} c_i s_i.$$

This reveals something remarkable: *the support of dual codewords determines reconstruction*. Specifically, if there exists a dual codeword c with $c_0 \neq 0$ and $c_i = 0$ for all $i \notin I$, then the coalition indexed by I can reconstruct the secret using only their shares. This observation leads to Massey's characterization theorem, which provides a complete correspondence between qualified sets and dual codewords.

Theorem 3.2.2 (Massey's Characterization)**.** [16] A coalition $I \subseteq \{1, \ldots, n\}$ is qualified if and only if there exists a codeword $c = (c_0, c_1, \ldots, c_n) \in \mathcal{C}^\perp$ with $c_0 \neq 0$ and

$$\textit{supp}(c) \subseteq \{0\} \cup I.$$

Moreover, after normalizing so that $c_0 = 1$, the secret can be reconstructed via

$$s = -\sum_{i \in I} c_i s_i.$$

Proof. [16] (*Forward direction*) Suppose there exists a codeword $c = (c_0, c_1, \ldots, c_n) \in \mathcal{C}^\perp$ with $c_0 \neq 0$ and $\textit{supp}(c) \subseteq \{0\} \cup I$. For any valid codeword $y = (s, s_1, \ldots, s_n) \in \mathcal{C}$, we have

$$0 = c \cdot y = c_0 s + \sum_{i \in I} c_i s_i,$$

since $c_i = 0$ for all $i \notin I$. As $c_0 \neq 0$, this equation can be solved for s, showing that the coalition I can reconstruct the secret and is therefore qualified.

(*Reverse direction*) Suppose $I \subseteq \{1, \ldots, n\}$ is qualified. By Theorem 3.2.1, this means that

$$g_0 \in \operatorname{span}\{g_i : i \in I\}.$$

Hence there exist coefficients $\lambda_i \in \mathbb{F}_q$ such that

$$g_0 = \sum_{i \in I} \lambda_i g_i.$$

Define a vector $c = (c_0, c_1, \ldots, c_n) \in \mathbb{F}_q^{n+1}$ by

$$c_0 = 1, \qquad c_i = \begin{cases} -\lambda_i, & \text{if } i \in I, \\ 0, & \text{if } i \notin I. \end{cases}$$

Then $\textit{supp}(c) \subseteq \{0\} \cup I$ by construction. For any message vector $u \in \mathbb{F}_q^k$ and the corresponding codeword $y = uG \in \mathcal{C}$, we obtain

$$c \cdot y = c_0(u \cdot g_0) + \sum_{i \in I} c_i (u \cdot g_i) = u \cdot \Big(g_0 - \sum_{i \in I} \lambda_i g_i \Big) = 0.$$

Thus $c \in \mathcal{C}^\perp$ with $c_0 = 1$ and $\textit{supp}(c) \subseteq \{0\} \cup I$, completing the proof. □

This theorem establishes that the access structure of a secret sharing scheme is completely determined by the support patterns of codewords in the dual code. But we can say even more: the *minimal* qualified sets correspond precisely to *minimal* codewords.

Corollary 3.2.3 (Minimal Qualified Sets via Minimal Codewords)**.** [16] A qualified set I is minimal if and only if there exists a minimal codeword $c \in \mathcal{C}^\perp$ with $c_0 = 1$ and

$$\textit{supp}(c) = \{0\} \cup I.$$

Proof. Suppose I is minimal qualified and let $c \in \mathcal{C}^\perp$ be the corresponding dual codeword from Theorem 3.2.2. If c were not minimal, there would exist another non-zero codeword $c' \in \mathcal{C}^\perp$ with $\textit{supp}(c') \subsetneq \textit{supp}(c)$. But then the dual code would contain a codeword with support strictly contained in $\{0\} \cup I$, implying that a proper subset of I is qualified, contradicting the minimality of I. Conversely, if c is a minimal codeword with the stated properties, any proper subset $I' \subsetneq I$ cannot be qualified, since that would produce a dual codeword with support strictly contained in $\textit{supp}(c)$, contradicting the minimality of c. □

Example 3.2.2 (Example of Corollary 3.2.3)**.** Let $\mathcal{C}$ be an $[4, 2]$-linear code over $\mathbb{F}_5$ with generator matrix

$$G = \begin{bmatrix} g_0 \mid g_1 \mid g_2 \mid g_3 \end{bmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

A dual codeword $c = (c_0, c_1, c_2, c_3) \in \mathcal{C}^{\perp}$ satisfies $c_0 + c_1 + c_3 = 0$ and $c_2 + c_3 = 0$.

For $c_0 = 1$, two minimal choices are

$$c = (1, 4, 0, 0) \quad \text{and} \quad c = (1, 0, 1, 4),$$

with supports $\{0, 1\}$ and $\{0, 2, 3\}$, respectively. By Corollary 3.2.3, these correspond to the minimal qualified sets $\{P_1\}$ and $\{P_2, P_3\}$.

Although minimal codewords provide an exact characterization of minimal qualified sets, their computation is algorithmically difficult. Vardy [22] showed that computing the minimum distance of a linear code is NP-hard, and enumerating minimal codewords is at least as hard. Ashikhmin and Barg [2] studied the structure and number of minimal codewords, showing that for general linear codes, the number of minimal codewords can be exponential in the code length. This computational barrier has significant implications for secret sharing: since minimal codewords of the dual code characterize minimal qualified sets (Corollary 3.2.3), directly computing the access structure from a code is generally intractable. This explains why practical LSSS constructions—including the optimal and threshold-based approaches studied in this thesis—avoid explicit enumeration of minimal codewords and instead rely on algebraic or structural methods.

Construction 1 and Construction 2 demonstrate how linear codes naturally induce secret sharing schemes. The key insight from Massey's characterization (Theorem 3.2.2) is that minimal codewords of the dual code precisely determine the access structure. This observation raises a natural question: *given a desired access structure, can we systematically construct a linear code whose dual has exactly the correct minimal codewords?*

Part II

Approaches for LSSS Construction

Part II addresses this question by presenting two complementary approaches. Chapter 4 develops the *optimal approach*, which searches for codes of minimal length by solving algebraic constraint systems derived from the access structure. When a solution exists, this approach yields an ideal LSSS with the smallest possible share size. Chapter 5 presents the *threshold-based approach*, which constructs LSSS matrices directly from threshold access trees—the tree representations introduced in Section 2.2. This approach handles (t, n)-threshold gates natively and runs efficiently even for large access structures, though it may not always produce schemes of minimal size. The two approaches reflect a fundamental trade-off: the optimal approach guarantees minimality but is computationally expensive, while the threshold-based approach is efficient but may sacrifice optimality. In the practical part of this thesis, we compare both methods on concrete examples and evaluate when each is preferable.

4
The Optimal Approach

Construction 2 (Section 3.2) relates access structures to dual codewords, but it is not constructive because computing the relevant minimal codewords in $\mathcal{C}^{\perp}$ is NP-hard in general.

In this chapter we follow the *optimal approach* of Tang, Gao, and Zhang [9]. Their key idea is to reduce the existence of a linear code realizing a given access structure to the solvability of a system of polynomial equations over $\mathbb{F}_q$. Concretely, from the collection of *minimal* qualified sets $\Gamma = \{S_1, \ldots, S_m\}$ one builds a matrix $H \in \mathbb{F}_q^{m \times (n+1)}$ whose i-th row has support exactly $\{0\} \cup S_i$. From the family $R = \{R_1, \ldots, R_\ell\}$ of maximal rejected sets (the adversary structure) one builds a matrix $G \in \mathbb{F}_q^{\ell \times (n+1)}$ whose i-th row has zeros precisely in the positions indexed by R_i. The access structure Γ is realizable by a linear code over $\mathbb{F}_q$ if and only if the constraint system

$$GH^{\top} = 0$$

has a solution, subject to the condition that all entries of H corresponding to participants in S_i are nonzero [9].

In the following, we proceed as follows. First, we explain the main idea of the optimal approach: how access structures can be encoded as linear codes and what "optimal" means in this context. Secondly, we will show how to derive the adversary structure from a given access structure and why maximal forbidden sets play a central role. Next, we will describe the construction of the matrices G and H and the constraint system $GH^{\mathsf{T}} = 0$. Section 4.1 presents the algorithm for solving this system and finding optimal codes. Finally, Section 4.1.3 discusses the advantages and limitations of this approach, particularly in comparison to the threshold-based method considered in Section 5. Recall from Chapter 3 that a subset $S \subseteq \{1, \ldots, n\}$ is qualified if and only if

$$g_0 \in \operatorname{span}\{g_i : i \in S\},$$

where $G = [\, g_0 \mid g_1 \mid \cdots \mid g_n \,] \in \mathbb{F}_q^{k \times (n+1)}$ is a generator matrix of a linear code $\mathcal{C} \subseteq \mathbb{F}_q^{n+1}$. The first column g_0 corresponds to a secret s, while the remaining columns represent the participants' shares. The goal of the optimal approach is to construct a matrix G such that:

$$\begin{aligned} S \in \Gamma &\iff g_0 \in \operatorname{span}\{g_i : i \in S\}, \\ S \notin \Gamma &\iff g_0 \notin \operatorname{span}\{g_i : i \in S\}. \end{aligned}$$

Among all generator matrices satisfying these relationships, the *optimal* one corresponds to a linear code of minimum possible length realizing the access structure. Minimizing the code length is equivalent to minimizing the total share size, and hence maximizing the information rate of the resulting linear secret–sharing scheme. In particular, when the minimum length equals $n + 1$, the scheme is *ideal*, otherwise, the optimal scheme is non-ideal but achieves the smallest possible share overhead among all linear realizations of the given access structure [9].

4.1 From Access Structures to Adversary Structures

Let $\Gamma = \{S_1, \ldots, S_m\}$ be a family of minimal qualified sets. In order to construct an optimal linear secret–sharing scheme, we must determine the associated family of maximal forbidden sets (the adversary structure). Since a set of participants is forbidden precisely when it fails to contain any minimal qualified set, it is convenient to describe forbidden coalitions in terms of how they intersect the family Γ. The notion of an *overlaying set* provides a combinatorial tool for this purpose: it characterizes those sets of participants that intersect every minimal qualified set, and hence allows maximal forbidden sets to be identified in a systematic way.

Definition 4.1.1 (Overlaying [9])**.** Let $\Gamma = \{S_1, \ldots, S_m\}$ be the family of minimal qualified sets and let $H = (1\ \Gamma) \in \mathbb{F}_q^{m\times(n+1)}$ be the matrix whose i-th row encodes S_i (i.e., $h_{ij} \neq 0$ iff $j \in S_i$). For $B \subseteq [1, m]$ and $T \subseteq [1, n]$, we say that T *overlays* B if for every $i \in B$ there exists $j \in T$ with $h_{ij} \neq 0$.

Theorem 4.1.1 (Characterization of Maximal Forbidden Sets [9])**.** A set $R \subseteq P$ is a maximal forbidden set if and only if its complement $T = P \setminus R$ is a minimal overlaying set.

Proof. Assume R is a maximal forbidden set. Then, it holds $R \notin \Gamma$, so for every minimal qualified set $S \in \Gamma_{\min}$, we must have $S \not\subseteq R$ as otherwise R would be qualified by monotonicity. This means that S contains at least one element which is not in R, i.e., $S \cap (P \setminus R) \neq \emptyset$. Thus $T = P \setminus R$ overlays Γ. Minimality of T follows from maximality of R. Conversely, if T is a minimal overlaying set, then $R = P \setminus T$ cannot contain any minimal qualified set (otherwise T would not intersect that set). Thus R is forbidden. Maximality follows from minimality of T. □

As a conclusion of this Theorem, the adversary structure $\mathcal{R}_{\max}$ can be obtained by enumerating all minimal overlaying sets T and taking complements. This step is combinatorial and may be costly for large access structures. Nevertheless, it provides a complete description of the forbidden coalitions.

Construction of H and G and their Relation Before we start with the building of the matrices, we need to remark that in the optimal approach, the access structure is represented such that *participants correspond to columns rather than rows* of the resulting matrix. Each column encodes the share contribution of a participant, while the rows correspond to constraints derived from the access structure.

This column-oriented representation is inherent to the underlying algorithm on which the optimal approach is based. In this thesis, the algorithm is adopted as given, and no attempt is made to alter or transpose the construction in order to obtain a row-oriented representation. The resulting matrix is mathematically equivalent to row-oriented constructions used in threshold-based schemes, but it leads to different matrix dimensions with practical implications.

In the optimal approach, two matrices determine the realization of a given access structure as a linear secret-sharing scheme. We provide here an intuitive overview of the roles of H and G; detailed constructions are given in the subsection 4.1.1 and 4.1.2.

- **Matrix** H encodes the *access structure* $\Gamma = \{S_1, \ldots, S_m\}$. Each row of H corresponds to one minimal qualified set S_i, with non-zero entries indicating which participants belong to that set.

- **Matrix** G encodes the *adversary structure* $\mathcal{R}_{\max} = \{R_1, \ldots, R_\ell\}$. Each row of G corresponds to one maximal forbidden set R_i, with zero entries indicating which participants belong to that forbidden set [9].

The central constraint of the optimal approach is the orthogonality relation

$$G\,H^{\mathsf{T}} = 0.$$

This condition enforces both security and correctness simultaneously:

- **Privacy**: Each row of G corresponds to a maximal forbidden set $R_i \subseteq \{1, \ldots, n\}$. The structure of G ensures that participants in R_i cannot reconstruct the secret, because no codeword in the dual code with first coordinate 1 has support contained within R_i.
- **Reconstruction**: Each row of H corresponds to a minimal qualified set S_i. The orthogonality condition ensures that these rows lie in the dual code, enabling participants in S_i to reconstruct the secret.

This construction mirrors the generator/parity-check duality from coding theory [12, 20]: if a solution to $GH^{\mathsf{T}} = 0$ exists, then the row space of G defines a linear code $\mathcal{C}$, and the rows of H lie in the dual code $\mathcal{C}^{\perp}$. A solution to this system yields an ideal LSSS realizing the given access structure Γ.

Real Adversary Structure In practice, the theoretical adversary structure $\mathcal{R}$ derived from the access structure is often significantly larger than the family of coalitions that are actually relevant for enforcing privacy. While $\mathcal{R}$ contains all unqualified sets, only the *maximal* forbidden coalitions are needed to describe the privacy constraints of the optimal construction. These maximal sets forming $\mathcal{R}_{\max}$ represent the real adversaries in the sense that every smaller unqualified coalition is contained in at least one of them and therefore does not impose additional restrictions on the matrices G and H. Working with $\mathcal{R}_{\max}$ rather than the full adversary structure is essential: it yields the minimal collection of privacy constraints that any linear secret sharing scheme realizing the given access structure must satisfy, and it avoids an exponential proliferation of redundant constraints. Consequently, the constraint matrix H is constructed solely from $\mathcal{R}_{\max}$, and the relation $GH^{\mathsf{T}} = 0$ captures precisely the privacy conditions induced by these real adversaries.

4.1.1 Construction of Matrix H

Once the minimal qualified sets $S_1, \ldots, S_m$ are determined from the access structure Γ, we construct the matrix H whose rows encode the reconstruction constraints. For each minimal qualified set S_i, the corresponding row h_i of H must enable participants in S_i to reconstruct the secret. Concretely, H is an $m \times (n+1)$ matrix of the form

$$H = \begin{pmatrix} 1 & h_{11} & h_{12} & \cdots & h_{1n} \\ 1 & h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & h_{m1} & h_{m2} & \cdots & h_{mn} \end{pmatrix},$$

where:

- The first column (the "secret column") consists entirely of 1s.
- For each row i and participant $j \in \{1, \ldots, n\}$:

  - $h_{ij} \neq 0$ if $j \in S_i$ (participant j belongs to the qualified set S_i),
  - $h_{ij} = 0$ if $j \notin S_i$.
- The non-zero entries h_{ij} (for $j \in S_i$) are treated as unknowns over $\mathbb{F}_q$.

4.1.2 Construction of Matrix G

Once the maximal forbidden sets $R_1, \ldots, R_\ell$ are determined from the adversary structure $\mathcal{R}_{\max}$, we construct the matrix G whose rows encode the privacy constraints. The idea is that for every forbidden set R_i, participants in R_i must not be able to reconstruct the secret. By the following Lemma,

Lemma 4.1.2 (Characterization of maximal rejected sets [9, Lemma 4])**.** Let Γ be a monotone access structure on the participant set $P = \{1, \ldots, n\}$, and let $R \subseteq P$. Then R is a maximal forbidden (rejected) set if and only if there exists a vector

$$c = (c_0, c_1, \ldots, c_n) \in \mathbb{F}_q^{n+1}$$

such that:

1. $c_0 = 1$,
2. $c_j = 0$ for all $j \in R$,
3. $c_j \neq 0$ for all $j \in P \setminus R$.

So we know that it is equivalent to requiring the existence of a codeword in the code $\mathcal{C}$ with the first coordinate 1 and zeros in all positions corresponding to R_i. Concretely, G is an $\ell \times (n+1)$ matrix of the form

$$G = \begin{pmatrix} 1 & g_{11} & g_{12} & \cdots & g_{1n} \\ 1 & g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & g_{\ell 1} & g_{\ell 2} & \cdots & g_{\ell n} \end{pmatrix},$$

where:

- The first column consists entirely of 1s.
- For each row i and participant $j \in \{1, \ldots, n\}$:
  - $g_{ij} = 0$ if $j \in R_i$ (participant j belongs to the forbidden set R_i),
  - g_{ij} is an unknown if $j \notin R_i$.

The zero pattern in each row of G ensures that the corresponding forbidden set cannot use the reconstruction formula from Equation (2.3.2): since $g_{ij} = 0$ for all $j \in R_i$, only participants outside of R_i contribute to any linear combination involving row g_i.

Example 4.1.1 (Example of a Matrix H and G)**.** Consider participants $\{1, 2, 3\}$ with $\Gamma_{\min} = \big\{\{1, 2\}, \{1, 3\}\big\}$ and $\mathcal{R}_{\max} = \big\{\{2, 3\}\big\}$, and let h and $g \in \mathbb{F}_q$. Then:

$$H = \begin{pmatrix} 1 & h_{11} & h_{12} & 0 \\ 1 & h_{21} & 0 & h_{23} \end{pmatrix}, \qquad G = \begin{pmatrix} 1 & g_{11} & 0 & 0 \end{pmatrix}.$$

Creating the Linear Secret Sharing Scheme

Each maximal forbidden set $R \in \mathcal{R}_{\max}$ imposes a privacy constraint: the participants in R must not be able to reconstruct the secret. Equivalently, the secret column g_0 must not lie in the span of the columns $\{g_i : i \in R\}$. By the construction of the matrices G and H, all such constraints can be encoded simultaneously by the linear system

$$G\,H^{\mathsf{T}} = 0.$$

The equation $G\,H^{\mathsf{T}} = \mathbf{0}$ captures both requirements of a linear secret sharing scheme in a single algebraic condition:

- *Privacy:* for every maximal forbidden set $R \in \mathcal{R}_{\max}$, the corresponding row of G enforces that no linear combination of the shares held by R yields the secret.
- *Reconstruction:* for every minimal qualified set $S \in \Gamma$, the corresponding row of H ensures that $g_0 \in \operatorname{span}\{g_i : i \in S\}$.

Consequently, solving the system $G\,H^{\mathsf{T}} = 0$ over $\mathbb{F}_q$ yields a linear secret sharing scheme that realizes the access structure Γ, while failure of the system implies that no such realization exists with the prescribed matrix dimensions.

Theorem 4.1.3 (Existence of Linear Realizations [9])**.** Let Γ be a monotone access structure on a participant set $P = \{1, \ldots, n\}$, and let $\mathcal{R}_{\max}$ be the associated family of maximal forbidden sets. Construct the constraint matrix H from $\mathcal{R}_{\max}$ as described above. Then, there exists a linear secret–sharing scheme realizing Γ over$\mathbb{F}_q$ with underlying generator matrix $G \in \mathbb{F}_q^{k\times(n+1)}$ if and only if the system

$$GH^{\top} = 0$$

admits a solution over $\mathbb{F}_q$.

Proof. [9] Let Γ be a monotone access structure on $P = \{1, \ldots, n\}$, and let $\mathcal{R}_{\max}$ denote the family of maximal forbidden sets. By construction, the constraint matrix H has one row for each $R \in \mathcal{R}_{\max}$ and encodes the linear conditions imposed by these forbidden sets.

Let

$$G = (g_0, g_1, \ldots, g_n) \in \mathbb{F}_q^{k\times(n+1)}$$

be a generator matrix, where the first column corresponds to the secret and the remaining n columns correspond to the participants' shares. A set $R \subseteq P$ is unqualified if and only if the secret is independent of the shares held by R, which is equivalent to $g_0 \notin \langle g_i : i \in R\rangle$. By linear duality, this holds if and only if there exists a nonzero vector h_R supported on $\{0\} \cup R$ such that

$$Gh_R^{\top} = 0.$$

Collecting one such vector h_R for each $R \in \mathcal{R}_{\max}$ as a row yields precisely the matrix H. Hence, the condition that every maximal forbidden set is unqualified is equivalent to the system

$$GH^{\top} = 0.$$

Conversely, suppose that $GH^{\top} = 0$ admits a solution over $\mathbb{F}_q$. Then for every $R \in \mathcal{R}_{\max}$ there exists a dual vector witnessing that R is unqualified, and by monotonicity every subset of R is also unqualified. For any qualified set $A \notin \mathcal{R}_{\max}$, no such constraint exists, which implies that $g_0 \in \langle g_i : i \in A\rangle$ and thus the secret can be recovered as a linear combination of the shares of A.

The integer k is the dimension of the underlying linear code, and the optimal construction is obtained by choosing k minimal among all solutions of $GH^{\top} = 0$; if $k = 1$, i.e., the code length is exactly $n + 1$, the resulting scheme is ideal. □

The optimal approach searches for a solution of the system $GH^\top = 0$ with the smallest possible number of columns. The first dimension at which a solution exists yields an *optimal* linear secret–sharing scheme. If this dimension equals $n + 1$, the resulting scheme is *ideal*.

Example 4.1.2 (Concrete system of equations)**.** Consider three participants $P = \{1, 2, 3\}$ with minimal qualified sets

$$\Gamma_{\min} = \big\{\{1, 2\}, \{1, 3\}\big\}$$

and maximal forbidden sets

$$\mathcal{R}_{\max} = \big\{\{2, 3\}\big\}.$$

Let $\mathbb{F}_q$ be a finite field with $q > 2$.

Following the construction of the optimal approach, we build the matrices H and G as follows. For each minimal qualified set S_i, the corresponding row of H has first coordinate 1 and nonzero entries exactly in the positions indexed by S_i:

$$H = \begin{pmatrix} 1 & h_{11} & h_{12} & 0 \\ 1 & h_{21} & 0 & h_{23} \end{pmatrix}, \quad h_{11}, h_{12}, h_{21}, h_{23} \in \mathbb{F}_q.$$

For the unique maximal forbidden set $\{2, 3\}$, the corresponding row of G has zeros in positions 2 and 3:

$$G = \begin{pmatrix} 1 & g_{11} & 0 & 0 \end{pmatrix}, \quad g_{11} \in \mathbb{F}_q.$$

The constraint $G\,H^{\mathsf{T}} = 0$ yields the system

$$\begin{cases} 1 + g_{11}h_{11} = 0, \\ 1 + g_{11}h_{21} = 0. \end{cases}$$

This system admits a solution if and only if $h_{11} = h_{21}$ and $g_{11} = -h_{11}^{-1}$ in $\mathbb{F}_q$. Since $h_{11} \neq 0$ by construction, such a solution always exists over any field $\mathbb{F}_q$ with $q > 2$.

Consequently, the system $G\,H^{\mathsf{T}} = 0$ is solvable, and hence there exists an ideal linear secret–sharing scheme realizing the access structure $\Gamma_{\min}$.

The Main Algorithm Tang, Gao, and Zhang [9] provide an algorithm that systematically searches for optimal linear secret sharing schemes by iteratively attempting to solve the constraint system $GH^{\mathsf{T}} = 0$ for codes of increasing length until a solution is found.

The algorithm operates in two phases. In the first phase, it attempts to find an *ideal* LSSS of length $n + 1$ (one coordinate for the secret plus one for each participant). If no ideal scheme exists, the second phase searches for *non-ideal* schemes by systematically increasing the code length.

Phase 1: Searching for an Ideal LSSS

Given an access structure Γ on n participants, the algorithm first attempts to construct an ideal LSSS as follows:

1. **Compute minimal qualified sets.** Extract $\Gamma_{\min} = \{S_1, \ldots, S_m\}$ from the access structure Γ.
2. **Compute maximal forbidden sets.** Using Theorem 4.1.1, determine $\mathcal{R}_{\max} = \{R_1, \ldots, R_\ell\}$ by finding minimal overlaying sets and taking complements.
3. **Construct matrix H.** Build an $m \times (n+1)$ matrix where row i corresponds to the minimal qualified set S_i. The first column contains all 1s (representing the secret coordinate). For $j \in \{1, \ldots, n\}$: if participant $j \in S_i$, then H_{ij} is an unknown variable over $\mathbb{F}_q$; otherwise $H_{ij} = 0$.

4. **Construct matrix G.** Build an $\ell \times (n+1)$ matrix where row i corresponds to the maximal forbidden set R_i. The first column contains all 1s. For $j \in \{1, \ldots, n\}$: if participant $j \in R_i$, then $G_{ij} = 0$; otherwise G_{ij} is an unknown variable.

5. **Solve the constraint system.** Attempt to find values for all unknown variables such that $GH^{\mathsf{T}} = 0$ over $\mathbb{F}_q$.

   - If a solution exists, instantiate G and H with the solution values. The rows of H form a parity-check matrix for the desired code, yielding an ideal LSSS of length $n + 1$.
   - If no solution exists, proceed to Phase 2.

Phase 2: Searching for Non-Ideal LSSS

When no ideal LSSS exists, the algorithm extends the matrices by adding columns, effectively increasing the code length. The key insight is that adding columns corresponds to allowing participants to hold multiple share components.

Column extension principle. When extending the matrices from $n + 1$ columns to $n + 1 + k$ columns, the new columns must preserve the zero-pattern constraints that encode the access structure. Specifically, if the original column j has a zero in row i of G (because participant j belongs to forbidden set R_i), then any new column "associated with" participant j must also have a zero in row i. We say two columns have the *same form* if they share identical zero patterns in both G and H.

The extension proceeds iteratively:

1. For $k = 1, 2, 3, \ldots$:

   (a) Construct extended matrices G_k and H_k with $n + 1 + k$ columns by appending k new columns to G and H.

   (b) Each new column copies the zero pattern of one existing participant column (columns 2 through $n + 1$), but introduces fresh unknown variables for the non-zero entries.

   (c) Attempt to solve $G_k H_k^{\mathsf{T}} = 0$.

   (d) If a solution exists, return the corresponding LSSS of length $n + 1 + k$.

Example (Phase 2: Non-Ideal LSSS). Let $P = \{P_1, P_2, P_3\}$ and consider the access structure

$$\Gamma = \{S \subseteq P \mid \{P_1, P_2\} \subseteq S \text{ or } \{P_1, P_3\} \subseteq S\}.$$

Assume that no ideal LSSS exists for Γ.

The minimal qualified and maximal forbidden sets are

$$\Gamma_{\min} = \big\{\{P_1, P_2\}, \{P_1, P_3\}\big\}, \qquad \mathcal{R}_{\max} = \big\{\{P_2, P_3\}, \{P_1\}\big\}.$$

Initial matrices

With column order $(\text{secret}, P_1, P_2, P_3)$, the matrices constructed in Phase 1 are

$$H = \begin{pmatrix} 1 & h_{1,1} & h_{1,2} & 0 \\ 1 & h_{2,1} & 0 & h_{2,3} \end{pmatrix}, \qquad G = \begin{pmatrix} 1 & g_{1,1} & 0 & 0 \\ 1 & 0 & g_{2,2} & g_{2,3} \end{pmatrix}.$$

By assumption, the system $GH^{\mathsf{T}} = 0$ has no solution.

Column extension

We now add one column associated with participant P_2, copying the zero pattern of the P_2-column. Ordering columns as $(\text{secret}, P_1, P_2, P_3, P_2')$, the extended matrices are

$$H_1 = \begin{pmatrix} 1 & h_{1,1} & h_{1,2} & 0 & h_{1,4} \\ 1 & h_{2,1} & 0 & h_{2,3} & 0 \end{pmatrix}, \qquad G_1 = \begin{pmatrix} 1 & g_{1,1} & 0 & 0 & 0 \\ 1 & 0 & g_{2,2} & g_{2,3} & g_{2,5} \end{pmatrix}.$$

The system $G_1 H_1^{\mathsf{T}} = 0$ is then solved over $\mathbb{F}_q$. A solution yields a non-ideal LSSS of length 5, where participant P_2 receives two share components, illustrating the column extension principle of Phase 2.

The formal algorithm is presented below.

Input
  Access structure Γ on participant set $\mathcal{P} = \{1, \ldots, n\}$
  Finite field $\mathbb{F}_q$

Output
  Generator Matrix G realizing Γ with minimal length, or failure

Phase 1: Ideal LSSS
  Compute $\Gamma_{\min}$ and $\mathcal{R}_{\max}$ from Γ
  Construct initial matrices G and H with $n + 1$ columns
  if $GH^{\top} = 0$ has a solution over $\mathbb{F}_q$ **then**
    return the G of length $n + 1$

Phase 2: Non-Ideal LSSS
  $k \leftarrow 1$
  repeat
    Construct G_k and H_k by appending k columns to G and H
    Each new column has the same zero pattern as some existing column $j \in \{2, \ldots, n + 1\}$
    if $G_k H_k^{\top} = 0$ has a solution over $\mathbb{F}_q$ **then**
      return the LSSS of length $n + 1 + k$
    $k \leftarrow k + 1$
  until k exceeds maximum iterations

Algorithm 1: Construction of an Optimal Linear Code Realizing Γ [9].

Proposition 4.1.1 (Termination and Optimality)**.** If the algorithm terminates with a solution at length $n + 1 + k$, then:

1. The resulting LSSS correctly realizes the access structure Γ.
2. No LSSS over $\mathbb{F}_q$ realizing Γ has length smaller than $n + 1 + k$.

Proof. Correctness follows from Theorem 4.1.3: the constraint $GH^{\mathsf{T}} = 0$ ensures that qualified sets can reconstruct the secret (their corresponding rows of H lie in the dual code) while forbidden sets cannot (their corresponding rows of G enforce the necessary zero patterns). Optimality follows from the iterative search: since the algorithm checks lengths $n + 1, n + 2, \ldots$ in order, the first successful length is minimal. □

The computational bottleneck lies in solving the polynomial system $GH^{\mathsf{T}} = 0$. Over $\mathbb{F}_2$, all non-zero entries of H equal 1, reducing the system to linear equations solvable by Gaussian elimination in polynomial time. Over larger fields, the system is quadratic in the unknowns $\{h_{ij}, g_{ij}\}$, and solving it may

require exponential time in the worst case. Additionally, computing $\mathcal{R}_{\max}$ via minimal overlaying sets can be expensive when $|\Gamma_{\min}|$ is large, since determining all minimal overlaying sets is a combinatorial problem and may require enumerating an exponential number of candidates in the worst case.

We will see that this is difficult for some access structures.

Example 4.1.3. Let $\mathcal{P} = \{p_1, \ldots, p_{100}\}$ be a set of participants and consider an access structure $\Gamma = \{\{p_1, p_2\}\}$. When we now want to generate the adversary structure $\mathcal{R}$ we see that this will have many sets in it. Since any set missing at least one of $\{p_1, p_2\}$ is unqualified, the two maximal unqualified sets are

$$P \setminus \{p_1\} = \{p_2, p_3, \ldots, p_{100}\} \quad \text{and} \quad P \setminus \{p_2\} = \{p_1, p_3, \ldots, p_{100}\}.$$

Hence, the maximal adversary structure is

$$\mathcal{R}_{\max} = \big\{P \setminus \{p_1\},\, P \setminus \{p_2\}\big\}.$$

The complete adversary structure consists of all subsets of these maximal sets, which contains $2^{99} + 2^{99} - 2^{98} = 3 \cdot 2^{98}$ sets in total.

Computing $\mathcal{R}_{\max}$ via overlaying sets can be expensive for access structures with many minimal qualified sets. This limits the practical applicability of the optimal approach to moderately-sized access structures.

4.1.3 Advantages and Limitation

The optimal approach has several notable advantages and some significant limitations.

Advantages

1. **Theoretical optimality**: When an ideal LSSS exists, the optimal approach is guaranteed to find it. The resulting scheme achieves the minimal possible share size (one field element per participant).
2. **Generality**: The method applies to arbitrary monotone access structures, not just threshold policies or structured formulas.
3. **Algebraic foundation**: The approach is grounded in the well-understood theory of linear codes and provides a clear mathematical framework for understanding LSSS construction.
4. **Completeness**: If the constraint system has no solution, we can conclude definitively that no ideal LSSS exists for the given access structure over the chosen field.

Limitations

1. **Computational cost of adversary structure**: Computing $\mathcal{R}_{\max}$ via overlaying sets can be expensive. For access structures with many minimal qualified sets or complex overlapping patterns, this step may become prohibitive.
2. **Solving polynomial systems**: Over fields larger than $\mathbb{F}_2$, the constraint system $GH^{\mathsf{T}} = 0$ is quadratic in the unknowns. Solving such systems efficiently is challenging, and no general polynomial-time algorithm is known.
3. **Field dependence**: The existence of a solution may depend on the choice of field $\mathbb{F}_q$. A structure that has no ideal LSSS over $\mathbb{F}_2$ might have one over $\mathbb{F}_5$, requiring multiple attempts.

4. **No direct support for threshold trees**: If the access structure is naturally expressed as a threshold tree, the optimal approach requires first converting it to minimal qualified sets, then computing adversary sets, then solving constraints multiple expensive steps.

5. **Scalability**: The approach is most effective for small to medium-sized access structures (up to $\sim$10-20 participants). For larger structures, both the adversary computation and system solving become expensive.

5
The Threshold-Based Approach

The optimal approach of Chapter 4 constructs LSSS by solving algebraic constraint systems derived from access and adversary structures. While theoretically elegant and guaranteed to find minimal share sizes when they exist, this approach faces computational challenges: computing overlaying sets can be expensive, and solving polynomial systems does not always scale well to large access structures. In practice, many access structures arise from hierarchical or threshold-based policies. Examples include:

- Organizational approval policies: ”2 directors and 1 manager” or ”3 out of 5 board members”
- Blockchain quorum sets: nested threshold conditions like those in Stellar.[17]
- Attribute-based encryption: policies like ”((Manager AND Finance) OR (Director AND HR)) AND Legal”

For such structures, there exists an alternative construction method that works directly on the tree representation, without explicitly computing adversary structures or solving polynomial systems. This *threshold-based approach*, introduced by Liu, Cao, and Wong [14], builds LSSS matrices *top-down* from threshold access trees, handling (t, n)-threshold gates natively.

This chapter proceeds as follows. First, we explain the core idea: how to assign share-generating vectors to tree nodes and preserve span conditions across gates. Secondly, we will present the recursive construction for OR, AND, and general (t, n)-threshold gates. Next, we will give the complete algorithm. Additionally, we will prove correctness and analyze the matrix size. Finally, last but not least, we discuss advantages, limitations, and the comparison to the optimal approach.

The Idea Recall from Definition 2.3.5 that a linear secret sharing scheme for a monotone access structure is specified by a pair (M, ρ), where $M \in \mathbb{F}_q^{\ell \times d}$ is the share-generating matrix and $\rho : \{1, \ldots, \ell\} \to \mathcal{P}$ labels each row by a participant. A set of participants $S \subseteq \mathcal{P}$ is qualified if and only if the rows of M labeled by S span the target vector $\mathbf{e}_1 = (1, 0, \ldots, 0) \in \mathbb{F}_q^d$; otherwise, $\mathbf{e}_1$ is not in their span, and the set cannot reconstruct the secret.

The threshold-based approach takes as input a monotone Boolean formula (Definition 2.2.4) represented as a *threshold access tree*, a tree whose internal nodes are (t, n)-threshold gates as defined in Definition 2.2.5.

Recall that AND gates correspond to (n, n)-thresholds (all inputs required) and OR gates correspond to $(1, n)$-thresholds (any single input suffices). The algorithm constructs the pair (M, ρ) in a *top-down* manner, assigning a *share-generating vector* to each node and combining child contributions via structured linear transformations that preserve the span condition at every gate. Crucially, general (t, n)-threshold nodes are handled directly, no expansion into large conjunctions and disjunctions of AND/OR gates is required [14].

From Threshold Trees to LSSS Construction Outline Let each node v in the access tree be associated with a linear map that transforms its parent vector into row-vectors for the subtree of v. The construction proceeds recursively:

1. **Root** Start with the target vector $e_1 = (1)$. This is the "seed" label at the root.
2. **Leaf** If v is a leaf labeled by attribute A, emit one row of M equal to the current vector $u_v \in \mathbb{F}_q^d$ and set $\rho(\text{row}) = A$.
3. **OR ($t = 1$)** For an OR-node with children $v_1, \ldots, v_n$ and parent vector u, *propagate* u to each child unchanged: $u_{v_i} \leftarrow u$. (Intuitively, any one child suffices to span u.)
4. **AND ($t = n$)** For an AND-node with children $v_1, \ldots, v_n$ and parent vector $u \in \mathbb{F}_q^d$, *grow the dimension* by 1. Let $d' \leftarrow d + 1$. Define

$$u_{v_1} \leftarrow (0, \ldots, 0, -1) \in \mathbb{F}_q^{d'}, \qquad u_{v_2}, \ldots, u_{v_n} \leftarrow \text{independent extensions so that } u = \sum_{i=1}^{n} u_{v_i}.$$

   This enforces that all n children are jointly needed to reconstruct u.

5. (t, n)**-threshold.** Generalize the AND/OR patterns: choose a *local* share-combining matrix $T \in \mathbb{F}_q^{n \times r}$ with r small, so that any t rows of T span a fixed vector (serving as u), but any $< t$ rows do not. Map u through T to obtain child vectors u_{v_i}; recurse on each child with u_{v_i}. Over $\mathbb{F}_q$ with $q > n$, a compact choice for T is obtained via Vandermonde-style[11] rows, ensuring MDS behaviour at the gate.

By design, a set of leaves satisfies the node exactly when their rows collectively span the parent vector. Composing gates top–down preserves the overall span condition at the root, hence realizes the whole access structure as an LSSS.[14]

Correctness and Size *Correctness* is by induction on the tree: each node guarantees that its authorized subsets of children span the parent vector while unauthorized subsets do not. Thus, a leaf set satisfies the root if and only if it satisfies every ancestor gate, i.e., if and only if it satisfies the intended policy.

Size The number of LSSS rows equals the number of leaves. Unlike AND/OR-only methods that first expand threshold node into large monotone formulas, the native (t, n) handling avoids the formula blow-up, yielding smaller matrices and hence smaller ciphertexts in LSSS-based CP-ABE. In the worst case (only AND/OR gates), the size matches prior methods; for threshold-heavy trees, it is strictly better.[14]

Example: 2-out-of-4 Threshold Consider a $(2, 4)$-threshold gate over participants $\{A, B, C, D\}$. Over $\mathbb{F}_5$, we construct the share-generating matrix using distinct evaluation points $\alpha_i \in \{1, 2, 3, 4\}$:

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \qquad \rho(1) = A,\ \rho(2) = B,\ \rho(3) = C,\ \rho(4) = D.$$

Any two rows $(1, \alpha_i)$ and $(1, \alpha_j)$ with $\alpha_i \neq \alpha_j$ are linearly independent and thus span the target vector $\mathbf{e}_1 = (1, 0)$. No single row does, since $(1, \alpha_i)$ with $\alpha_i \neq 0$ cannot be scaled to $(1, 0)$. Hence, exactly the 2-out-of-4 threshold condition is realized. This construction generalizes: for any (t, n)-threshold gate, a matrix with t columns ensures that any t rows are linearly independent.

Algorithm for Converting Threshold Trees to LSSS Matrices

The following algorithm formally converts a general threshold access tree into an LSSS matrix. It generalizes the well-known Lewko–Waters [13] conversion by supporting threshold gates directly rather than requiring their expansion into AND/OR subtrees.

Algorithm 1 Generate LSSS from a threshold access tree [14]

Require: Threshold access tree $\mathcal{T}$, finite field $\mathbb{F}_q$ (with $q > n$)
Ensure: LSSS matrix M and labeling function ρ

Initialize empty list of rows $M \leftarrow [\,]$ and labeling map $\rho \leftarrow \emptyset$
Assign the root node the label vector $v_{\text{root}} \leftarrow (1)$
procedure EXPANDNODE(x, v_x)
    if x is a leaf labeled by attribute A **then**
        Append v_x as a new row of M
        $\rho(\text{row}) \leftarrow A$
    else if x is an OR gate (1-of-n) **then**
        for each child y of x **do**
            EXPANDNODE(y, v_x)
        end for
    else if x is an AND gate (n-of-n) **then**
        Extend v_x by one dimension: $v'_x \leftarrow (v_x, 0)$
        for each child y_i of x **do**
            Assign $v_{y_i} \leftarrow e_i$-based extension of v'_x
            EXPANDNODE(y_i, v_{y_i})
        end for
    else ▷ x is a (t, n)-threshold gate
        Construct local threshold matrix $T \in \mathbb{F}_q^{n \times t}$ such that any t rows are linearly independent
        for each child y_i of x **do**
            $v_{y_i} \leftarrow v_x \cdot T_i$ ▷ T_i: i-th row of T
            EXPANDNODE(y_i, v_{y_i})
        end for
    end if
end procedure
EXPANDNODE(root,(1))
return (M, ρ)

Algorithm Description The procedure EXPANDNODE traverses the threshold tree recursively. Each node receives a vector v_x whose dimension grows only when necessaryspecifically, at AND or general (t, n) gates. The root starts with $v = (1)$, and each recursive call generates new row-vectors for child nodes. When a leaf is reached, the algorithm appends the vector to M and records the attribute label ρ. The resulting matrix has one row per leaf (i.e., per attribute occurrence in the tree), and the number of columns equals the final vector dimension reached along the deepest path.

Advantages The threshold-tree-based construction natively supports (t, n)-threshold gates without requiring an expansion into equivalent AND/OR formulas. This is particularly important for practical applications such as blockchain quorum sets, where threshold conditions arise naturally and compact representations are essential. From a computational perspective, the algorithm is efficient: it runs in $O(\ell)$ time, where ℓ denotes the number of leaves in the tree. In contrast to optimal constructions based on adversary structures, no computation of minimal rejected sets or solving systems of polynomial equations is required. This leads to a predictable and easily analyzable performance profile. Another practical benefit is the predictability of the resulting LSSS matrix size. The number of rows is exactly equal to the number of leaves in the threshold tree, which allows the size of the scheme to be estimated precisely before construction. This property is particularly useful in resource-constrained environments. The approach also scales well to large access structures. Even threshold trees containing hundreds of leaves can be processed efficiently, making the method suitable for large-scale policies. Furthermore, the construction is conceptually simple and straightforward to implement. It relies only on basic linear algebra, specifically vector assignments derived from threshold gates, and avoids symbolic equation solving entirely. Finally, the method provides a direct representation of access policies that are naturally expressed as threshold trees. In such cases, no intermediate conversion into minimal qualified sets is necessary, which simplifies both the conceptual model and the implementation.

Limitations Despite its advantages, the threshold-tree-based approach does not guarantee optimality. In particular, it does not necessarily minimize the share size. For certain access structures, the resulting LSSS may require more columns than an optimal construction obtained via code-based methods. The construction also imposes requirements on the size of the underlying field. For a (t, n)-threshold gate, it is necessary that $q \geq n$ in order to ensure the existence of n distinct field elements for the Vandermonde-based construction. As a result, gates with many children may require working over comparatively large fields. Another limitation is the growth of the vector dimension. Each AND or threshold gate introduces additional dimensions, and for deeply nested trees this effect can accumulate. While the growth is typically modest in practice, it can become noticeable for highly structured policies. Moreover, the access structure must be provided explicitly as a threshold tree, or be convertible into one. For access structures that are naturally specified as collections of qualified sets, this conversion step may be nontrivial and can introduce additional complexity. In contrast to the optimal approach, the threshold method offers no guarantee of ideality. While it always produces a valid LSSS, it cannot determine whether a smaller or ideal scheme exists for the same access structure. Finally, attribute repetition may increase the matrix size. If the same attribute appears multiple times in different branches of the threshold tree, it will correspond to multiple rows in the LSSS matrix. Although this correctly captures the semantics of the policy, it can lead to a larger-than-necessary representation.

This chapter has presented the threshold-based approach for constructing LSSS directly from threshold access trees. The key advantage is computational efficiency: the construction runs in linear time and handles (t, n)-gates natively without formula expansion. The trade-off is that the resulting LSSS may not be optimal in terms of share size. In the next part, we implement both the optimal and threshold-based approaches and compare their performance on concrete examples, including access structures derived from the Stellar blockchain.

Part III

Practical Part

6
Practical Part

6.1 Material

In this part, we apply the two constructions developed in the theoretical part to concrete access structures, including examples derived from the Stellar blockchain [18]. To this end, we implemented a Python script that takes an access structure as input and automatically produces a corresponding LSSS. The implementation supports both the *optimal approach* and the *threshold-based approach*. We proceed as follows. First, we give a brief overview of Stellar and explain how validator quorum sets naturally induce monotone threshold access structures. In particular, we describe how Stellar's nested quorum sets can be translated into the formal access structures used throughout this thesis. Next, we present the implementation of the optimal approach. Starting from a given access structure, we explain how we derive the associated adversary structure and how we construct the constraint families $\mathcal{H}$ and $\mathcal{G}$. We then show how these objects are used to obtain an LSSS matrix by solving the system

$$GH^\top = 0.$$

This procedure is illustrated on three access structures: two synthetic examples and one access structure originating from Stellar. Subsequently, we turn to the threshold-based approach. We show how to represent access structures as threshold access trees and how our implementation converts such trees into LSSS matrices. Again, we illustrate the process on a tree-based toy example and on a Stellar-derived access structure. Finally, we compare the two approaches in practice. We discuss the resulting matrix sizes, the structure of the generated schemes, the computational effort of the constructions, and summarize the advantages and limitations of both approaches in a practical setting.

Implementation Environment All implementations are written in Python. We rely on the standard library modules `argparse` and `typing` for command-line handling and type annotations, and on `numpy` for efficient linear-algebraic operations.

Input Representation of Access Structures We work with two equivalent input formats for an access structure. First, we allow an input given by an enumeration of list. From this representation we either

construct the adversary structure, which is used directly in the optimal approach, or we derive an AND OR term that can be transformed into a threshold tree for the threshold approach. Second, we support a threshold-tree representation that can be used directly by the threshold approach and that can also be converted back into a family of qualified sets for the optimal approach. Each node of the threshold tree is encoded as a tuple of the form

$$(P_1, P_2, \ldots, P_m, t),$$

where $P_1, \ldots, P_m$ are either participants or subtrees, and t is the threshold. Intuitively, at least t of the listed children must be satisfied in order to satisfy the node. For example, the tuple $(A, B, 1)$ expresses that either $\{A\}$ or $\{B\}$ is sufficient to achieve the quorum. If only a single participant is required, we still use the same notation and write, for instance, $(A, 1)$. Our parser also supports nested threshold systems. For example,

$$((A, B, 1), (C, D, 1), 1)$$

describes two sub-threshold systems $(A, B, 1)$ and $(C, D, 1)$, of which at least one has to be satisfied.

Test Cases and Datasets

We evaluate both implementations on two synthetic access structures:

Test Case 1: Two-level OR-AND Structure

$$\Gamma_1 = ((A, B, C, 1), (D, E, 1), 2)$$

This structure requires:

- Either: one of $\{A, B, C\}$ AND one of $\{D, E\}$
- Minimal qualified sets: $\{A, D\}, \{A, E\}, \{B, D\}, \{B, E\}, \{C, D\}, \{C, E\}$

Test Case 2: Nested Threshold Structure

$$\Gamma_2 = ((A, B, 2), (C, (A, E, F, 1), 2), 1)$$

This structure requires:

- Either: both A AND B
- Or: both C AND one of $\{A, E, F\}$
- Minimal qualified sets: $\{A, B\}, \{A, C\}, \{C, E\}, \{C, F\}$

Test Case 3: Stellar

As a real-world case study, we apply our constructions to access structures arising in the Stellar blockchain. Unlike blockchains based on proof of work or classical proof of stake, Stellar uses the Stellar Consensus Protocol (SCP), which is an instance of a Federated Byzantine Agreement (FBA) consensus protocol[17]. In SCP, each validator specifies a *quorum set*, consisting of a set of trusted validators together with a threshold that determines how many of them must agree. These quorum sets may be nested and combined, and thus naturally give rise to monotone threshold access structures of the kind studied in this thesis. To obtain concrete examples, we use the OBSRVR Radar service[18], which exposes the quorum configuration of individual Stellar validators. From this data we extract the corresponding access structures (or threshold access trees) and use them as input for both the optimal approach and the threshold-based approach to generate and compare the resulting LSSS matrices.

Stellar Consensus and Quorum Sets In the Stellar network, each node is associated with a nested quorum set. We can extract these quorum configurations for individual nodes using the OBSRVR Radar tool [18]. Conceptually, the access tree for a node is organized in two layers. On the top level, the node specifies a collection of trusted organizations. The node requires agreement from a certain number of these organizations in order to accept a statement, which naturally yields a threshold condition over organizations. On the second level, each organization defines its own internal quorum set: it selects a number of its validator nodes that must agree before the organization itself contributes a positive vote.
Thus, the access structure of a Stellar node can be modeled as a nested threshold tree, a top-level threshold over trusted organizations, where each organization is represented by a lower-level threshold over its validator nodes. An example of such an access tree is illustrated in Figure 2.2.5. We will use this as a test case for our approach.

6.2 Implementation of the Optimal Approach

As we have seen in Chapter 4, the optimal approach transforms the problem of constructing an LSSS into a problem of solving a constrained linear algebra system. The fundamental insight is that an access structure can be realized by a linear code if and only if certain orthogonality conditions are satisfied between the generator matrix and the dual code. These conditions manifest as the matrix equation $GH^T = 0$, where G encodes the adversary structure and H encodes the access structure.

Our implementation follows a sequential pipeline that mirrors the theoretical development. Starting from an access structure represented either as a collection of minimal qualified sets or as a threshold access tree, we first ensure that we have an explicit representation of the minimal qualified sets. From these, we construct the incidence matrix H that captures which participants belong to which qualified sets. In parallel, we compute the adversary structure, identifying the maximal sets of participants that should not be able to reconstruct the secret. This adversary structure then determines the structure of the matrix G. Finally, we formulate and solve the polynomial system $GH^T = 0$ to obtain the actual LSSS matrix, provided such a solution exists over the chosen finite field.

The implementation is designed to handle access structures over finite fields $\mathbb{F}_q$ where q is a prime. For the binary field $\mathbb{F}_2$, the polynomial system reduces to a linear system, which can be solved efficiently using Gaussian elimination. For larger prime fields, the system remains quadratic in the unknowns, requiring more sophisticated symbolic or numerical techniques.

Construction of matrices H and G Starting from the given threshold description, we first parse it into *groups* of participants together with a global threshold. From this description we algorithmically generate all *minimal qualified sets*

$$\Gamma = \{S_1, S_2, \ldots, S_m\}$$

by enumerating, for each group, all local subsets that meet the local group threshold and then combining groups until the global threshold is satisfied. Each such combination that cannot be made smaller without losing the property of being qualified is stored as a minimal qualified set. From the collection Γ we then construct the matrix H as a binary incidence matrix between qualified sets and participants: rows correspond to the sets S_i, columns correspond to participants, and

$$H_{ij} = \begin{cases} 1 & \text{if participant } j \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

After we get the access structure Γ and the matrix H, we construct a linear secret sharing scheme in the form of a monotone span program. We work over a finite field $\mathbb{F}_q$ (for a small prime p) and build a matrix

$$G \in \mathbb{F}_q^{k \times (n+1)},$$

where n is the number of participants and the first column represents the secret component. Conceptually, the remaining columns encode randomness in such a way that the rows corresponding to any qualified set $S \in \Gamma$ span the target vector

$$\mathbf{e}_1 = (1, 0, \ldots, 0),$$

while the rows corresponding to any adversary qualified set do not. Concretely, the code assigns to each participant a columns of G, sets the first column entries so that the secret appears with coefficient 1 in every share, and fills the remaining columns with simple field elements chosen so that the rows of each minimal qualified set form a linear system whose solution expresses $\mathbf{e}_1$ as a linear combination of these columns. Afterwards, the implementation verifies correctness by checking that the rows indexed by each qualified set indeed span $\mathbf{e}_1$ and that, for every adversary set, $\mathbf{e}_1$ is *not* in their column span.

Solving the Constraint System and Constructing the LSSS For a fixed column length L and a fixed column-to-participant mapping, the construction of a linear secret sharing scheme is reduced to solving a system of linear constraints over a finite field. All computations in the implementation are performed over the prime field $\mathbb{F}_{17}$. The construction proceeds in two stages. First, a candidate matrix H is instantiated. The matrix H has one row for each minimal authorized set $S \in \Gamma$ and one column for each share (including the secret column). The first column, corresponding to the secret, is fixed to 1 in every row. For each authorized set S, all columns whose participant labels belong to S are assigned independent, uniformly random nonzero elements of $\mathbb{F}_{17}$, while all other entries are fixed to 0. This randomized instantiation respects the structural constraints implied by the access structure while avoiding degenerate solutions caused by zero coefficients. Once H is fixed, the orthogonality condition

$$GH^\top = 0$$

induces a system of linear equations for the unknown entries of the matrix G. The matrix G has one row for each maximal rejected set $R \in \mathcal{R}$. As for H, the first column of G is fixed to 1 in every row. For a rejected set R, all columns owned by participants in R are fixed to 0; the remaining entries are treated as unknown variables. For each rejected row, orthogonality with every authorized row of H yields a homogeneous linear system over $\mathbb{F}_{17}$. These linear systems are solved independently for each rejected row using Gaussian elimination over $\mathbb{F}_{17}$. If a solution exists for all rejected rows, the resulting matrix G satisfies the global orthogonality condition $GH^\top = 0$, and the pair (G, H) forms a candidate LSSS for the given access structure. If no solution exists for at least one rejected row, the current instantiation of H is discarded. Because the entries of H are chosen randomly, the feasibility of the linear systems is probabilistic. The implementation therefore repeats the sampling of H and the subsequent linear solving for G up to a user-defined trial limit. If no feasible solution is found within this bound, the algorithm concludes that no scheme exists for the current column length and duplication pattern and proceeds to the next configuration. Whenever a solution is found, the resulting scheme is explicitly verified: the orthogonality condition is re-checked, all minimal authorized sets are tested for successful reconstruction, and all maximal rejected sets are tested for failure. Only schemes passing all verification steps are accepted and reported as valid linear secret sharing schemes.

Example 6.2.1 (Optimal Approach $((A, B, C, 1), (D, E, 1), 2)$)**.** This approach is straightforward. First, we generate the minimal qualified sets. The following 6 minimal qualified sets are obtained:

$$\begin{aligned} Q_1 &= \{A, D\}, \\ Q_2 &= \{A, E\}, \\ Q_3 &= \{D, B\}, \\ Q_4 &= \{B, E\}, \\ Q_5 &= \{C, D\}, \\ Q_6 &= \{C, E\}. \end{aligned}$$

From these we can derive the minimal adversary (maximal rejected) sets:

$$R_1 = \{D, E\},$$
$$R_2 = \{A, B, C\}.$$

$$H = \begin{pmatrix} 1 & h_{0,1} & 0 & 0 & h_{0,4} & 0 \\ 1 & h_{1,1} & 0 & 0 & 0 & h_{1,5} \\ 1 & 0 & h_{2,2} & 0 & h_{2,4} & 0 \\ 1 & 0 & h_{3,2} & 0 & 0 & h_{3,5} \\ 1 & 0 & 0 & h_{4,3} & h_{4,4} & 0 \\ 1 & 0 & 0 & h_{5,3} & 0 & h_{5,5} \end{pmatrix}, \qquad G = \begin{pmatrix} 1 & 0 & 0 & 0 & g_{0,4} & g_{0,5} \\ 1 & g_{1,1} & g_{1,2} & g_{1,3} & 0 & 0 \end{pmatrix}.$$

For the ideal column length $L = 6$, the column-to-participant mapping is given by

$$\text{Column } 0 : [\text{secret}], \quad \text{Column } 1 : A, \quad \text{Column } 2 : B, \quad \text{Column } 3 : C, \quad \text{Column } 4 : D, \quad \text{Column } 5 : E.$$

After applying these constraints, the concrete matrix $H \in \mathbb{F}_{17}^{6\times 6}$ is

$$H = \begin{pmatrix} 1 & 2 & 0 & 0 & 3 & 0 \\ 1 & 2 & 0 & 0 & 0 & 7 \\ 1 & 0 & 11 & 0 & 3 & 0 \\ 1 & 0 & 11 & 0 & 0 & 7 \\ 1 & 0 & 0 & 5 & 3 & 0 \\ 1 & 0 & 0 & 5 & 0 & 7 \end{pmatrix}.$$

Solving for G over $\mathbb{F}_{17}$ then gives:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 11 & 12 \\ 1 & 8 & 3 & 10 & 0 & 0 \end{pmatrix}.$$

We can verify that $G * H^t = 0$ We can also verify the matrix with two examples.

Example 6.2.2 (Optimal Approach $((A, B, 2), (C, (A, E, F, 1), 2), 1)$)**.** This yields the following 4 minimal qualified sets:

$$Q_1 = \{A, B\},$$
$$Q_2 = \{A, C\},$$
$$Q_3 = \{C, E\},$$
$$Q_4 = \{C, F\}.$$

From these, we obtain the maximal rejected sets:

$$R_1 = \{A, E, F\},$$
$$R_2 = \{B, E, F\},$$
$$R_3 = \{B, C\}.$$

The successful configuration duplicates participant A by adding an extra share column:

$$\text{Column } 0 : [\text{secret}], \ \text{Column } 1 : A, \ \text{Column } 2 : B, \ \text{Column } 3 : C, \ \text{Column } 4 : E, \ \text{Column } 5 : F, \ \text{Column } 6 : A.$$

With this mapping, the symbolic matrices become:

$$H = \begin{pmatrix} 1 & h_{0,1} & h_{0,2} & 0 & 0 & 0 & h_{0,6} \\ 1 & h_{1,1} & 0 & h_{1,3} & 0 & 0 & h_{1,6} \\ 1 & 0 & 0 & h_{2,3} & h_{2,4} & 0 & 0 \\ 1 & 0 & 0 & h_{3,3} & 0 & h_{3,5} & 0 \end{pmatrix}, \qquad G = \begin{pmatrix} 1 & 0 & g_{0,2} & g_{0,3} & 0 & 0 & 0 \\ 1 & g_{1,1} & 0 & g_{1,3} & 0 & 0 & g_{1,6} \\ 1 & g_{2,1} & 0 & 0 & g_{2,4} & g_{2,5} & g_{2,6} \end{pmatrix}.$$

From $GH^\top = 0$, the symbolic elimination yields only the consistency requirement

$$h_{1,3} = h_{2,3} = h_{3,3},$$

which is satisfiable. Hence, the expanded scheme is feasible. Instantiating coefficients in $\mathbb{F}_{17}$ and enforcing $h_{1,3} = h_{2,3} = h_{3,3} = 11$ gives the concrete matrix $H \in \mathbb{F}_{17}^{4\times 7}$:

$$H = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 5 \\ 1 & 7 & 0 & 11 & 0 & 0 & 13 \\ 1 & 0 & 0 & 11 & 3 & 0 & 0 \\ 1 & 0 & 0 & 11 & 0 & 7 & 0 \end{pmatrix}.$$

After substituting the already-determined G-variables from the orthogonality equations, four remaining linear equations are solved in $\mathbb{F}_{17}$, resulting in:

$$G = \begin{pmatrix} 1 & 0 & 11 & 3 & 0 & 0 & 0 \\ 1 & 9 & 0 & 3 & 0 & 0 & 3 \\ 1 & 16 & 0 & 0 & 11 & 12 & 7 \end{pmatrix}.$$

6.3 Implementation of the Threshold Access Tree Approach

The implementation of the threshold access-tree approach closely follows the pseudocode of Algorithm 1, CONVERT(FA), from Efficient Generation of Linear Secret Sharing Scheme Matrices from Threshold Approach [5]. The translation from pseudocode to Python is essentially direct; we only add comments and structure to make the control flow and the matrix operations explicit.

Working Example

Example 6.3.1 (Algorithm 1 Approach $((A, B, C, 1), (D, E, 1), 2)$)**.** Using Algorithm 1 (threshold-gate conversion) of Liu Cao Wong, the implementation returns the LSSS (M, ρ) over $\mathbb{F}_{17}$ with matrix

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}, \qquad \rho(1) = A,\ \rho(2) = B,\ \rho(3) = C,\ \rho(4) = D,\ \rho(5) = E.$$

Equivalently, the rows owned by each attribute are:

$$M_A = (1, 1),\ M_B = (1, 1),\ M_C = (1, 1),\ M_D = (1, 2),\ M_E = (1, 2).$$

Share generation Let the secret be $s \in \mathbb{F}_{17}$ and choose one fresh randomizer $r \in \mathbb{F}_{17}$. Define the sharing vector

$$\mathbf{v} = (s, r)^\top.$$

Each share is the row inner product $\lambda_i = M_i\mathbf{v}$, and the share for attribute X is $\lambda_X = M_X \cdot \mathbf{v}$. Concretely:

$$\lambda_A = s + r, \quad \lambda_B = s + r, \quad \lambda_C = s + r, \quad \lambda_D = s + 2r, \quad \lambda_E = s + 2r \qquad (\text{all in } \mathbb{F}_{17}).$$

Reconstruction from a qualified set. Consider the qualified set $S = \{A, D\}$. Let

$$M_S = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \quad (\text{rows of } A \text{ and } D).$$

We look for coefficients $(\omega_A, \omega_D) \in \mathbb{F}_{17}^2$ such that

$$\omega_A M_A + \omega_D M_D \;=\; (1, 0),$$

which is equivalent to $(M_S)^\top (\omega_A, \omega_D)^\top = (1, 0)^\top$. Solving over $\mathbb{F}_{17}$ gives

$$\omega_D = -1 \equiv 16 \pmod{17}, \qquad \omega_A = 2 \pmod{17}.$$

Therefore, for valid shares (λ_A, λ_D) we reconstruct

$$s \;=\; 2\lambda_A + 16\lambda_D \pmod{17}.$$

Calculation Take $s = 5$ and choose $r = 9$ in $\mathbb{F}_{17}$. Then

$$\lambda_A = s + r = 5 + 9 = 14, \qquad \lambda_D = s + 2r = 5 + 18 \equiv 5 + 1 = 6 \pmod{17}.$$

Reconstruction with $S = \{A, D\}$ yields

$$2\lambda_A + 16\lambda_D \;=\; 2 \cdot 14 + 16 \cdot 6 \;=\; 28 + 96 \;=\; 124 \equiv 5 \pmod{17},$$

which matches the original secret. For $S = \{A\}$ we only have the single row $M_A = (1, 1)$. There is no scalar ω such that $\omega(1, 1) = (1, 0)$, hence $(1, 0) \notin \mathrm{Im}(M_S^\top)$ and the secret is not reconstructible.

Example 6.3.2 (Algorithm 1 Approach $((A, B, 2), (C, (A, E, F, 1), 2), 1)$)**.** Using Algorithm 1 (Liu Cao Wong threshold-gate conversion), the tool outputs the LSSS (M, ρ) over $\mathbb{F}_{17}$ with

$$M \;=\; \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 2 \\ 1 & 0 & 2 \\ 1 & 0 & 2 \end{pmatrix}, \qquad \rho(1) = A,\ \rho(2) = B,\ \rho(3) = C,\ \rho(4) = A,\ \rho(5) = E,\ \rho(6) = F.$$

We note that A appears *twice* (two distinct rows), i.e., participant A receives two shares.

Share generation Let the secret be $s \in \mathbb{F}_{17}$ and choose two independent randomizers $r_1, r_2 \in \mathbb{F}_{17}$. Define

$$\mathbf{v} = (s, r_1, r_2)^\top.$$

Shares are computed as $\lambda_i = M_i \mathbf{v}$:

$$\lambda_{A,1} = s{+}r_1, \quad \lambda_B = s{+}2r_1, \quad \lambda_C = s{+}r_2, \quad \lambda_{A,2} = s{+}2r_2, \quad \lambda_E = s{+}2r_2, \quad \lambda_F = s{+}2r_2 \qquad (\bmod 17).$$

Reconstruction from a qualified set: $S = \{A, B\}$**.** Since A has two rows, the submatrix for S uses the three rows

$$M_S \;=\; \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 2 \\ 1 & 2 & 0 \end{pmatrix} \quad \text{(rows for } A\text{-share 1, } A\text{-share 2, and } B\text{)}.$$

We seek coefficients $(\omega_{A,1}, \omega_{A,2}, \omega_B) \in \mathbb{F}_{17}^3$ such that

$$\omega_{A,1} M_{A,1} + \omega_{A,2} M_{A,2} + \omega_B M_B \;=\; (1, 0, 0).$$

One valid solution over $\mathbb{F}_{17}$ is

$$\omega_{A,1} = 2, \qquad \omega_{A,2} = 0, \qquad \omega_B = -1 \equiv 16 \pmod{17}.$$

Hence $S = \{A, B\}$ reconstructs the secret via

$$s \;=\; 2\lambda_{A,1} + 16\lambda_B \pmod{17}.$$

(Here the second share of A is not needed, but it is still part of the scheme.)

Reconstruction Take $s = 5$ and choose $r_1 = 9$, $r_2 = 4$ in $\mathbb{F}_{17}$. Then

$$\lambda_{A,1} = 5 + 9 = 14, \qquad \lambda_B = 5 + 2 \cdot 9 = 23 \equiv 6 \pmod{17}.$$

Reconstruction yields

$$2\lambda_{A,1} + 16\lambda_B \;=\; 2 \cdot 14 + 16 \cdot 6 \;=\; 28 + 96 \;=\; 124 \equiv 5 \pmod{17},$$

which matches the original secret.

Calculation For $S = \{A\}$, even using both rows of A,

$$\mathrm{span}\{(1, 1, 0), (1, 0, 2)\}$$

cannot produce $(1, 0, 0)$, since forcing the second coordinate to 0 implies the first coordinate becomes 0 as well. Equivalently, $(1, 0, 0) \notin \mathrm{Im}(M_S^\top)$, so A alone cannot reconstruct.

Application to the Stellar SDF 1 Quorum Set

We now apply both approaches to the access structure derived from the SDF 1 validator node in the Stellar blockchain in December 2025, as illustrated in Figure 2.2.5. This access structure consists of 7 organizations, each containing 3 validator nodes, with a hierarchical threshold policy:

- **Top level**: 5-of-7 threshold over organizations
- **Organization level**: 2-of-3 threshold over validators within each organization

The 21 participants are distributed across the following organizations:

Organization	**Validators**
Blockdaemon	Validator 1, Validator 2, Validator 3
SDF	SDF 1, SDF 2, SDF 3
SatoshiPay	Iowa, Singapore, Frankfurt
Franklin Templeton	SCV 1, SCV 2, SCV 3
LOBSTR	1 (Europe), 2 (Europe), 5 (India)
Stellar Node	Alpha, Beta, Gamma
PublicNode	Hercules, Lyra, Boötes

Table 6.1: Organization structure of the SDF 1 quorum set

Attribute	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}
Blockdaemon Val. 1	1	1	1	1	1	1	0	0	0	0	0	0
Blockdaemon Val. 2	1	1	1	1	1	2	0	0	0	0	0	0
Blockdaemon Val. 3	1	1	1	1	1	3	0	0	0	0	0	0
SDF 3	1	2	4	8	16	0	1	0	0	0	0	0
SDF 1	1	2	4	8	16	0	2	0	0	0	0	0
SDF 2	1	2	4	8	16	0	3	0	0	0	0	0
SatoshiPay Iowa	1	3	9	10	13	0	0	1	0	0	0	0
SatoshiPay Singapore	1	3	9	10	13	0	0	2	0	0	0	0
SatoshiPay Frankfurt	1	3	9	10	13	0	0	3	0	0	0	0
FT SCV 1	1	4	16	13	1	0	0	0	1	0	0	0
FT SCV 3	1	4	16	13	1	0	0	0	2	0	0	0
FT SCV 2	1	4	16	13	1	0	0	0	3	0	0	0
LOBSTR 5 (India)	1	5	8	6	13	0	0	0	0	1	0	0
LOBSTR 2 (Europe)	1	5	8	6	13	0	0	0	0	2	0	0
LOBSTR 1 (Europe)	1	5	8	6	13	0	0	0	0	3	0	0
Gamma Node Val.	1	6	2	12	4	0	0	0	0	0	1	0
Alpha Node Val.	1	6	2	12	4	0	0	0	0	0	2	0
Beta Node Val.	1	6	2	12	4	0	0	0	0	0	3	0
Hercules	1	7	15	3	4	0	0	0	0	0	0	1
Lyra	1	7	15	3	4	0	0	0	0	0	0	2
Boötes	1	7	15	3	4	0	0	0	0	0	0	3

Table 6.2: LSSS matrix for the Stellar SDF 1 access structure over $\mathbb{F}_{17}$

6.3.1 Threshold-Based Approach

The threshold-based approach handles this nested structure naturally. Algorithm 1 processes the access tree top-down, assigning share-generating vectors to each node. The construction works over $\mathbb{F}_{17}$ and produces a 21×12 LSSS matrix M, shown in Table 6.2.

Structure of the Matrix The matrix exhibits a clear block structure that reflects the hierarchical nature of the access tree:

- **Columns c_0 through c_4**: These five columns encode the top-level 5-of-7 threshold condition. The entries follow a Vandermonde-like pattern with evaluation points $\alpha \in \{1, 2, 3, 4, 5, 6, 7\}$, one for each organization. Specifically, organization i receives the vector $(1, \alpha_i, \alpha_i^2, \alpha_i^3, \alpha_i^4)$ in these columns. Validators within the same organization share identical values in columns c_0 through c_4.
- **Columns c_5 through c_{11}**: Each of these seven columns corresponds to one organization's internal 2-of-3 threshold. Within each organization, the three validators receive values $\{1, 2, 3\}$ in their designated column and zeros elsewhere. This ensures that any two validators from the same organization can jointly satisfy their organization's threshold condition.

6.3.2 Optimal Approach: Computational Infeasibility

The optimal approach requires first computing the minimal qualified sets and the maximal adversary structure. For the Stellar access structure, this computation becomes prohibitively expensive.

Counting Minimal Qualified Sets A set of validators is minimally qualified if and only if:

1. Exactly 5 organizations are represented (the minimum required by the top-level threshold).
2. Within each of these 5 organizations, exactly 2 validators participate (the minimum required by each organization's threshold).

The number of minimal qualified sets is therefore:

$$|\Gamma_{\min}| = \binom{7}{5} \times \binom{3}{2}^5 = 21 \times 3^5 = 21 \times 243 = 5{,}103.$$

Computing the Adversary Structure The adversary structure computation presents the main computational bottleneck. By Theorem 4.1.1, maximal forbidden sets correspond to complements of minimal overlaying sets. A set T overlays $\Gamma_{\min}$ if it intersects every minimal qualified set.

For 5,103 minimal qualified sets over 21 participants, enumerating all minimal overlaying sets requires checking intersection conditions for each candidate subset. The number of maximal forbidden sets can be estimated as follows.

A maximal forbidden set R must fail the top-level threshold, meaning it contains validators from at most 4 organizations that each contribute at least 2 validators to R. A lower bound on $|\mathcal{R}_{\max}|$ considers sets where exactly 4 organizations each contribute all 3 validators:

$$|\mathcal{R}_{\max}| \geq \binom{7}{4} \times 3^4 = 35 \times 81 = 2{,}835.$$

The actual number is significantly larger when considering configurations where organizations contribute only 2 validators, or where more than 4 organizations participate but fewer than 5 meet their internal threshold.

Constraint System Size The constraint matrices have the following dimensions:

- $H \in \mathbb{F}_q^{5103 \times 22}$: one row per minimal qualified set, with columns for the secret coordinate plus 21 participants.
- $G \in \mathbb{F}_q^{|\mathcal{R}_{\max}| \times 22}$: one row per maximal adversary set.

The product GH^T produces a matrix of dimension $|\mathcal{R}_{\max}| \times 5103$, yielding approximately

$$|\mathcal{R}_{\max}| \times 5{,}103 \geq 2{,}835 \times 5{,}103 \approx 14.5 \times 10^6$$

polynomial constraints. While many of these constraints may be redundant or trivially satisfied, the sheer number of equations makes the system intractable.

Computational Bottlenecks Our implementation encountered the following obstacles:

1. **Minimal qualified set enumeration**: Generating all 5,103 sets requires $O\big(\binom{7}{5} \times \binom{3}{2}^5\big)$ operations. This was manageable.
2. **Overlaying set computation**: Finding minimal overlaying sets requires, in the worst case, checking each of the $2^{21} \approx 2.1 \times 10^6$ subsets of participants against all 5,103 qualified sets. Even with pruning strategies, the number of intersection checks is approximately

$$2^{21} \times 5{,}103 \approx 1.07 \times 10^{10}.$$

3. **Adversary structure size**: The optimal approach requires computing the *maximal* forbidden sets $\mathcal{R}_{\max}$, which we called the "real" adversary structure in Chapter 4. While this is substantially smaller than the full adversary structure $\mathcal{R}$ (which contains *all* forbidden sets), both are combinatorially large for the Stellar configuration:
    - The *full* adversary structure $\mathcal{R}$ contains every subset of participants that fails to satisfy the access condition. Since a set is qualified only if it includes at least 5 organizations each contributing at least 2 validators, the vast majority of the $2^{21} \approx 2.1 \times 10^6$ possible subsets are forbidden. A rough estimate yields $|\mathcal{R}| > 2 \times 10^6$ sets.
    - The *real* adversary structure $\mathcal{R}_{\max}$ contains only the maximal elements of $\mathcal{R}$. As computed above, $|\mathcal{R}_{\max}| \geq 2{,}835$, but this lower bound only accounts for configurations where exactly 4 organizations contribute all their validators. The true count is significantly higher when considering partial contributions and boundary cases.
4. **Matrix construction and solving**: With matrices containing thousands of rows and constraint systems with millions of polynomial equations, both memory requirements and solving time exceed practical limits. The symbolic manipulation of GH^{T} alone requires representing and simplifying millions of polynomial expressions.

After several hours of computation on standard hardware, our implementation failed to complete the adversary structure computation. This demonstrates that the optimal approach does not scale to real-world Stellar quorum configurations.

6.4 Evaluation

Having applied both the optimal and threshold-based to the Test Case 1, Test Case 2 and the Stellar SDF 1 quorum set, we now present a systematic comparison of the two methods.

6.4.1 Comparison Criteria

We evaluate both approaches along four dimensions:

1. **Matrix size**: The dimensions of the resulting LSSS matrix (M, ρ), which directly affects storage and communication costs in applications.
2. **Computational complexity**: The time and space required to construct the LSSS from the access structure specification.
3. **Scalability**: How the construction cost grows with the number of participants and the complexity of the access structure.
4. **Applicability**: Which types of access structures each approach handles naturally.

6.4.2 Results Summary

Table 6.3 summarizes the results obtained for the considered test cases using the optimal approach and the threshold-based approach.

Access Structure	Approach	Matrix Size	Status
$((A, B, C, 1), (D, E, 1), 2)$	Optimal	2×6	Success
	Threshold	5×2	Success
$((A, B, 2), (C, (A, E, F, 1), 2), 1)$	Optimal	3×7	Success
	Threshold	6×3	Success
Stellar SDF 1 (21 validators)	Optimal	—	Failed
	Threshold	21×12	Success

Table 6.3: Comparison of optimal and threshold-based approaches on selected test cases

6.4.3 Analysis

Matrix size Before comparing the matrix sizes, it is important to recall that the two approaches use different matrix representations. In the optimal approach, participants correspond to *columns* of the generator matrix, with an additional first column reserved for the secret. In contrast, in the threshold-based approach (monotone span programs), participants correspond to *rows* of the matrix. Therefore, matrix dimensions are not directly comparable without taking this difference into account. For the first access structure $((A, B, C, 1), (D, E, 1), 2)$, both approaches yield matrices of comparable overall size, as expected, since the underlying threshold structure is simple and fully disjoint. In the second test case $((A, B, 2), (C, (A, E, F, 1), 2), 1)$, we initially expected the optimal approach to produce a smaller matrix because the threshold gates are not disjoint. However, the initial constraint system for the optimal approach admits no solution at minimal length, and an extended matrix is required. As a result, the final matrix size is comparable to that of the threshold-based construction. For the Stellar SDF 1 access structure with 21 validators, the optimal approach fails to produce a solution. This is due to the large number of minimal qualified sets and maximal forbidden sets, which leads to a very large constraint system $GH^\top = 0$. The resulting number of variables and equations exceeds practical limits for solving the system. Consequently, no matrix is available for comparison in this case, whereas the threshold-based approach remains feasible and successfully produces a matrix.

Scalability The scalability difference is dramatic: The threshold approach handled the 21-participant Stellar structure in under 100 milliseconds, producing a 21×12 matrix. The optimal approach failed to complete after several hours due to the combinatorial explosion in adversary structure computation.

This confirms that the optimal approach is practical only for small access structures (approximately 10–15 participants), while the threshold approach scales to structures with dozens or even hundreds of participants.

Applicability he **optimal approach** applies to any monotone access structure specified as qualified sets, but its computational cost limits practical use to small instances. The **threshold approach** requires the access structure to be expressed as a threshold tree, which is natural for hierarchical policies like Stellar quorum sets but may require conversion for other structures.

6.4.4 Verification of Correctness

For both approaches, we verified correctness by testing reconstruction:

- For each minimal qualified set $S \in \Gamma_{\min}$, we checked that the rows of M labeled by S span the target vector $\mathbf{e}_1$.

- For sample forbidden sets, we verified that $\mathbf{e}_1$ is not in the span of the corresponding rows.

All tests passed for the matrices produced by both approaches on the synthetic examples. For the Stellar example, only the threshold approach produced a testable matrix, which also passed the tests.

6.4.5 Recommendations

Based on our evaluation, we provide the following practical recommendations:

Use the threshold approach when:

- The access structure is naturally expressed as a threshold tree
- Scalability is important
- The structure has more than approximately 15 participants
- Predictable matrix size is desired

Use the optimal approach when:

- Minimal share size is critical
- The access structure is small (fewer than 10 participants)
- The structure does not have a natural threshold tree representation
- Determining whether an ideal LSSS exists is important

For Stellar quorum sets and similar blockchain consensus structures, the threshold approach is the only practical choice due to the nested threshold structure and the number of validators involved.

7 Discussion and Future Work

7.1 Summary of Findings

This thesis has studied two complementary approaches for constructing linear secret sharing schemes for general monotone access structures. We now summarize the main findings.

Theoretical Contributions We presented a unified treatment of the relationship between linear codes and secret sharing, emphasizing Massey's characterization theorem (Theorem 3.2.2), which establishes that minimal codewords of the dual code correspond exactly to minimal qualified sets. This connection provides the theoretical foundation for the optimal approach. We formalized both constructions in Chapter 3:

- **Construction 1** distributes all coordinates of a codeword as shares, with the access structure determined by which column subsets span the target vector e_1.
- **Construction 2** reserves one coordinate for the secret, enabling the direct connection to dual code minimal codewords.

We also formalized the threshold-based construction, showing how access trees with (t, n)-threshold gates can be directly converted to LSSS matrices (Algorithm 1) without expanding thresholds into Boolean formulas. The construction preserves the span condition at each gate, ensuring correctness by induction on the tree structure.

Practical Contributions We implemented both approaches in Python and evaluated them on synthetic examples and real-world access structures from the Stellar blockchain. Our experiments revealed fundamental trade offs:

- The **optimal approach** produces schemes with minimal share size when solutions exist, but faces computational barriers for structures with many participants due to the cost of computing adversary structures and solving polynomial systems.
- The **threshold approach** scales efficiently to large access structures and handles nested threshold policies naturally, but does not guarantee minimal share size.

For the Stellar SDF 1 quorum set with 21 validators, only the threshold approach was practical, producing a 21×12 LSSS matrix in time. The optimal approach failed to complete adversary structure computation after several days.

Answers to Research Questions Returning to the research questions posed in Chapter 1:

RQ1: *How can we construct LSSS using linear codes?*

Linear codes provide a complete characterization of LSSS through Massey's theorem. An ideal LSSS exists if and only if the constraint system $GH^{\mathsf{T}} = 0$ has a solution, where G and H encode the adversary and access structures respectively. The dual code's minimal codewords precisely determine the minimal qualified sets.

RQ2: *How can we construct LSSS efficiently from threshold access trees?*

Algorithm 1 provides the precise procedure for generating LSSS matrices from threshold trees. The construction runs in linear time $O(\ell)$ where ℓ is the number of leaves, and produces one row per attribute occurrence in the tree. Threshold gates are handled natively using Vandermonde-style share vectors.

RQ3: *How do the two approaches compare?*

The approaches exhibit complementary strengths:

- The optimal approach guarantees minimal share size but has limited scalability.
- The threshold approach scales well (tested up to 21 participants, extendable to hundreds) but may produce larger schemes.
- For practical applications like Stellar quorum sets, the threshold approach is the only viable option.

7.2 Open Problems

Can the adversary structure be computed more efficiently for specific classes of access structures? The "real" adversary structure $\mathcal{R}(C)$ from Section IV of Tang et al. [9] provides some improvement for decomposable structures, but the computational cost remains prohibitive for large structures. Algorithmic improvements or approximation methods would extend the applicability of the optimal approach.

Lower Bounds on Share Size Given an access structure, what is the minimum possible share size achievable by any LSSS? While the optimal approach finds the minimum among ideal schemes, non-ideal schemes may achieve smaller shares at the cost of using multiple field elements per participant. Tighter lower bounds would help evaluate how close practical constructions come to the theoretical optimum.

Field Size Requirements Both approaches require the field size to exceed the number of children at each threshold gate (for the Vandermonde construction). For the threshold approach with a (t, n)-gate, we need $q \geq n$. Can this requirement be relaxed while maintaining efficiency? Alternative constructions using smaller fields would be valuable for resource-constrained applications.

7.3 Conclusion

This thesis has provided a comprehensive study of linear secret sharing scheme constructions for general monotone access structures, comparing two fundamentally different approaches grounded in coding theory and threshold access trees.

The *optimal approach* of Tang, Gao, and Zhang [9] offers theoretical elegance and guarantees minimal share size through the algebraic framework of linear codes. The constraint system $GH^{\mathsf{T}} = 0$ precisely characterizes when ideal LSSS exist, connecting access structure properties to minimal codewords of dual codes. However, the computational cost of adversary structure enumeration and polynomial system solving limits practical applicability to small access structures.

The *threshold-based approach* of Liu, Cao, and Wong [5] provides practical efficiency through direct construction from threshold access trees. By handling (t, n)-threshold gates natively without formula expansion, the algorithm achieves linear-time complexity and produces LSSS matrices of predictable size. This scalability makes it suitable for real-world applications, as demonstrated by our successful construction for the Stellar SDF 1 quorum set.

Our experimental comparison reveals a fundamental trade-off in LSSS construction: optimality versus efficiency. For applications requiring minimal share size in small settings, the optimal approach remains valuable. For large-scale deployments with hierarchical threshold policies, the threshold approach is essential.

The techniques considered in this thesis have immediate applications in distributed cryptography, blockchain consensus, and attribute-based encryption. As distributed systems continue to grow in complexity, efficient methods for realizing sophisticated access control policies become increasingly important. We hope this work contributes to the practical deployment of linear secret sharing schemes in real-world cryptographic systems.

Bibliography

[1] Carlos Aguilar-Melchor, Philippe Gaborit, Julien Lavauzelle, Adrien Hauteville, and Matthieu Rambaud. *Code-Based Secret Sharing Schemes*. World Scientific, 2022.

[2] A. Ashikhmin and A. Barg. Minimal vectors in linear codes. *IEEE Transactions on Information Theory*, 44(5):2010–2017, 1998.

[3] Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 11–46, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[4] Amos Beimel. Secret-sharing schemes for general access structures: An introduction. Technical Report 2025/518, Cryptology ePrint Archive, 2025. Accessed: 2026-01-20.

[5] Amos Beimel. Secret-sharing schemes for general access structures: An introduction. 2025.

[6] Josh Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO '88, Lecture Notes in Computer Science*, volume 403, pages 27–35, Berlin, Heidelberg, 1990. Springer.

[7] George R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference*, pages 313–317. AFIPS Press, 1979.

[8] Ernest F. Brickell. Some ideal secret sharing schemes. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 468–475, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[9] Tang Chunming, Gao Shuhong, and Zhang Chengli. The optimal linear secret sharing scheme for any given access structure. Cryptology ePrint Archive, Paper 2011/147, 2011.

[10] Cunsheng Ding and Arto Salomaa. Secret sharing schemes with nice access structures. *Fundamenta Informaticae*, 72(1–13):1–13, 2006.

[11] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

[12] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.

[13] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, pages 568–588, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[14] Zhen Liu, Zhenfu Cao, and Duncan S. Wong. Efficient generation of linear secret sharing scheme matrices from threshold access trees. Cryptology ePrint Archive, Paper 2010/374, 2010.

[15] Philip Martin. Introducing Coinbase's Open Source MPC Cryptography Library, March 26 2025. Accessed: 2026-01-20.

[16] James L. Massey. Minimal codewords and secret sharing. Technical report, Signal and Information Processing Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, CH-8092 Zürich, 1993. Discusses the role of minimal codewords in linear codes and their application in secret-sharing schemes.

[17] David Mazières. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 2016.

[18] OBSRVR. Obsrvr radar, 2025. Stellar network monitoring dashboard.

[19] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[20] Ron M. Roth. Linear codes. In *Introduction to Coding Theory*, chapter 2. Cambridge University Press, 2006.

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

[22] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

Declaration on the Use of AI Tools

During the preparation of this thesis, we used AI-based language models (ChatGPT, Claude by Anthropic, DeepL Write and a self-hosted qwen3-vl) for the purpose of improving language, grammar, and clarity of the written text, as well as for structuring and organizing ideas.