Secret Sharing and its variants, Matroids, Combinatorics

A thesis submitted to Indian Statistical Institute
in partial fulfillment of the thesis requirements for the degree of
Doctor of Philosophy in Computer Science

Author:
Shion Samadder Chaudhury

Supervisor:
Prof. Bimal Kumar Roy

Applied Statistics Unit
Indian Statistical Institute
203, B. T. Road, Kolkata,

To my Parents.

Acknowledgements

This thesis finally becomes a reality and takes its current shape with the kind support of many individuals. I would like to extend my sincere thanks to all those who have supported me through this journey.

My deepest gratitude goes first to my adviser Prof. Bimal Kumar Roy who expertly guided me through my doctoral studies and who shared the excitement of five years of discovery. His unwavering enthusiasm for secret sharing and matroids kept me constantly engaged with my research and motivated me to look beyond these topics for more combinatorial problems. He always gave me time for discussion from his busy schedule and his personal generosity helped make my time at ISI enjoyable and explore research problems with complete freedom. I hope to receive his guidance for many years to come.

Indian Statistical Institute, Kolkata has provided me with a very nurturing research environment. I would like to sincerely thank the professors of our division, the Applied Statistics Unit, Dr. Mridul Nandi, Prof. Subhamoy Maitra, Dr. Sushmita Ruj, Prof. Palash Sarkar, Prof. Tapas Samanta and Dr. Gautam Paul for their constant support, encouragement and the useful discussions I have had with them during the course of my PhD. Special appreciation extends to my immediate senior Dr. Debolina Ghatak whose mentoring and encouragement have been especially valuable and helped sustain a positive atmosphere to conduct research. I am glad to have colleagues like Diptendu Chatterjee, Anwesha Law , Sebati Ghosh and Pallavi Ghosh who have constantly supported and helped me in various ways through these years. It was a great pleasure to have association with my lab partners and juniors like Bishwajit Chakraborty, Suprita Talnikar, Avishek Majumdar, Samir Kundu and Anandarup Roy who always made my research life enjoyable.

It was a great experience to attend the courses of Prof. Subhamoy Maitra, Dr. Gautam Paul, Prof. Palash Sarkar, Prof. Sandip Das, Prof. Dipti Prasad Mukherjee and Prof. Bhargav Bhattacharya. Their teaching influenced me a lot. I am grateful to Indian Statistical Institute for providing all the facilities and support required for my research.

I would like to give special thanks to Prof. Kouichi Sakurai for hosting me at his department at Kyushu University, Japan and Dr. Sabyasachi Dutta of the Department of Computer Science, University of Calgary for their valuable collaboration, comments and revisions of my papers. Their early insights initiated the initial part of this the-

sis. Dr. Dutta has been a constant support. He was fundamental in supporting me throughout, specially during the stressful and difficult moments and helped me a lot during my stay at Japan.

I am thankful to all the anonymous reviewers for their invaluable comments and suggestions towards improving the quality of my thesis. I am also grateful to the PhD-DSC committee of Indian Statistical Institute, Kolkata for their continuous support, arranging regular evaluations of my research works and providing proper feedback.

Finally, I express my profound gratitude to my family. I would like to thank my relatives, cousins and friends for their endless love and support. Words cannot express how grateful I am to my parents whose love, blessings and sacrifices have sustained me this far. A special thanks to my wife for her patience and continuous encouragement.

Date: 20th December, 2020.

Contents

Chapter 1

Introduction

The main focus of this thesis is secret sharing. Secret Sharing is a very basic and fundamental cryptographic primitive. It is a method to share a secret by a dealer among different parties in such a way that only certain predetermined subsets of parties can together reconstruct the secret while some of the remaining subsets of parties can have no information about the secret. Secret sharing was introduced independently by Shamir [139] and Blakely [20]. What they introduced is called a *threshold secret sharing scheme*. In such a secret sharing scheme the subsets of parties that can reconstruct a secret are all those subsets whose cardinality is greater than a predetermined threshold. In a latter work by Ito, Saito and Nishizeki [93], secret sharing schemes were constructed where the subsets of parties who can reconstruct the secret did not have any concrete mathematical description.

1.1 Illustrative Example

Secret sharing schemes mirror a real life scenario. Consider the following situation :

- A wealthy man (the dealer) keeps his money in a locker.
- He has four children (parties/participants) and gives them keys such that : Atleast three of them has to co-operate (bring their keys together) to open the locker.
- None of the children can open the locker on their own.
- Even if two of them bring their keys together, still they cannot open the locker.
- The above mentioned condition is a description of a **(3-out-of-4) threshold secret**

sharing scheme.

1.2 Formal Definition

Before going into the work that has been done in this thesis, we take a look at the formal definition of secret sharing schemes. The definitions have been taken from the survey of Amos Beimel [15].

Definition 1. ***Access structure :*** *- For a set of parties $P = \{p_1, \ldots, p_n\}$, a collection of subsets $\mathcal{A} \subseteq 2^P$ is said to be monotone if, $B \in \mathcal{A}$ and $B \subseteq C \implies C \in \mathcal{A}$. An access structure $\mathcal{A}$ is a monotone collection of non-empty subsets of P. A set $A \in \mathcal{A}$, $A \subseteq P$ is called an authorized set and a set $A \notin \mathcal{A}$, $A \subseteq P$ is called an unauthorized set .*

Definition 2. ***Distribution Scheme :-*** *Given a domain of secrets K, a set of random strings R and domains of shares $K_1, \ldots, K_n$, a distribution scheme is a pair $\Sigma = \langle \Pi, \mu \rangle$ where μ is a probability distribution on R and Π is a mapping*

$$\Pi : K \times R \longrightarrow K_1 \times \ldots \times K_n.$$

Definition 3. ***Secret Sharing Scheme :-*** *A secret sharing scheme consists of a dealer, a set of parties $P = \{p_1, \ldots, p_n\}$ and an access structure $\mathcal{A}$ defined on P. The dealer distributes a secret $k \in K$ by choosing a random string $r \in R$ according to μ, computing a vector of shares $\Pi(k, r) = (s_1, \ldots, s_n)$ and privately communicating the shares s_i to party p_i. In addition, the following two conditions should be satisfied*

1. ***Correctness :-*** *Any authorized set of parties can reconstruct the secret. For*

each authorized set $B \in \mathcal{A}$, $B \subseteq P$, there is a reconstruction function $RECON_B$: $\prod_{i|p_i \in B} K_i \longrightarrow K$ such that for every secret $k \in K$,

$$Pr[RECON_B(\Pi(k,r)_B) = k] = 1$$

where $\Pi(k,r)_B$ is the restriction of $\Pi(k,r)$ to its B-entries.

2. ***Perfect Privacy :-*** *Every unauthorized set cannot learn any information about the secret in the information theoretic sense. For any set $F \notin \mathcal{A}$, for every two secrets $a, b \in K$ and for every possible vector of shares $\langle s_j \rangle_{p_j \in F}$:*

$$Pr[\Pi(a,r)_F = \langle s_j \rangle_{p_j \in F}] = Pr[\Pi(b,r)_F = \langle s_j \rangle_{p_j \in F}].$$

Definition 4. ***Information ratio :-*** *The information ratio of a secret sharing scheme is the ratio between the maximum length of the shares and the length of the secret.*

The above definitions can be relaxed and can require that correctness holds with very high probability and the statistical distance between $\Pi(a,r)_F$ and $\Pi(b,r)_F$ is small. These relaxations are used heavily in this work.

1.3 Main Goal of Secret Sharing

The share size in their construction of Ito et al.[93] was $\Theta(2^n/\sqrt{n})$, where n is the number of parties. The known constructions of secret sharing schemes for general access structures have information ratio $2^{O(n)}$ where n is the number of parties in the access

structure making these constructions impractical. Hence one of the most important goal of secret sharing is to reduce the share size. Proving/disproving the following conjecture is the most important problem of secret sharing.

Conjecture [15]:- For every $n \in \mathbb{N}$, there exists an $\epsilon > 0$ and an access structure with n parties such that every secret sharing scheme distributing secret among n parties has share size $2^{\epsilon n}$.

No major progress has been made in this area inspite of considerable efforts.

1.4 Main problems of this thesis

Next we look at the main focus points of this thesis.

1.4.1 Low-complexity Secret Sharing

In this section we consider secret sharing schemes where the share and the reconstruction procedures can be implemented using very low resources i.e., in a very low complexity class. The complexity class under consideration is the class AC^0 . This class consists of families of circuits of constant depth and polynomial size with unbounded fanin OR and AND gates. NOT gates are only allowed at the input.

Secret Sharing in AC^0

The motivation to construct secret sharing schemes where the secret sharing and the reconstruction procedures can be done in AC^0 comes from the following :-

1. General theme of randomized algorithms :- Improving performance at the price of allowing some small probability of error.

2. Reducing the computational complexity of cryptographic primitives.

3. The existing constructions of dynamic and evolving secret sharing schemes prior to our work not being AC^0 implementable.

In classical secret sharing the number of parties is fixed, the dealer knows the access structure beforehand and the access structure is also fixed. However in *dynamic secret sharing* schemes the addition or deletion of parties is allowed, the access structure might change with time and the dealer might go offline at a certain stage making the old parties virtual dealers. All these are done without reconstructing the secret. *Evolving secret sharing* schemes take one step further and allow an unbounded number of new parties. Parties come on by one and share is given to the new parties without any communication to the old parties. In a (k, ∞)- evolving scheme, at any time any k participants can reconstruct the secret. In a recently introduced generalization of such schemes, evolving schemes with increasing thresholds were constructed where the thresholds change with time and form an increasing sequence.

In our work we have obtained the following results in this respect [39] :

- Construction of an AC^0-implementable $(2, \infty)$-evolving scheme. The construction is very simple and is perhaps of theoretical interest only as the construction cannot be generalized to an AC^0-implementable (k, ∞)-evolving scheme.

- Since we are unable to construct (k, ∞)-evolving schemes accommodating unbounded number of parties, we look at AC^0 implementable dynamic schemes

where only a bounded number of new parties can be accommodated. The construction can be used in the dealer free situation and by simple modifications one can have AC^0 implementable dynamic schemes with hierarchical properties where some parties have preference over the others.

The main tools that are used in the constructions are functions which are AC^0 implementable such as *random permutations, approximate majority, good error-correcting codes, pseudorandom generators, k-wise independent generators* etc.

Secret Sharing via Dynamic Data Structures

In the constructions of evolving secret sharing schemes [99], [100] one finds an exponential amount of memory requirement. Also in the construction of the AC^0 implementable dynamic scheme [39], we can only accommodate a bounded number of new parties. To overcome these, we introduce a new variant of secret sharing schemes we call *Perpetual Secret Sharing* [40], [38] where the access structure evolves as per a *dynamic data structure*. This results in the following advantages :

1. The scheme can accommodate an unbounded number of parties.

2. Negligible memory requirements.

3. Share sizes do not increase drastically as new parties arrive.

4. When number of parties become very large, the data structure helps in efficient management of the parties like searching, addition and deletion of parties, pushing a party up or down a hierarchical order and so on.

1.4.2 Variants of Quantum Secret Sharing Schemes

Quantum Evolving Secret Sharing

Quantum secret sharing schemes to share and protect a quantum secret have been extensively studied. Numerous schemes have have been constructed which can add or delete participants. But prior to our work [37], a quantum secret sharing scheme which can accommodate an unbounded number of participants was not considered. In this direction we construct a *quantum evolving secret sharing* scheme. Our construction uses quantum one-time pad and trap codes to generate random states and also to mask the original quantum secret. The drawback of this construction is that the memory requirement is exponential.

Quantum Secret Sharing - Other variants

Some modifications to the construction of the quantum evolving scheme results in a *quantum hierarchical scheme* . Our constructed scheme has several advantages over the existing schemes in literature. Using similar ideas we are able to construct quantum versions of various *multipartite access structures*.

1.4.3 Embedding hard functions in secret sharing schemes

To obtain resilience in the secret sharing schemes, we embed a function with known lower bound on the *quantum query complexity* in the share state[37]. By resilience we mean that an adversary can share and modify a constant fraction of the shares and even then the secret can be reconstructed by the parties. We prove that compared to

coding theoretic methods, this results in reducing the dimension of the share state. This method also works in the classical case where we use functions with known lower bound on the randomized decision tree complexity . For our constructions the adversary is infinitely powerful/not resource bounded.

1.5 On Matroids and Non-perfect Secret Sharing

In the context of secret sharing, the family of subsets of participants who can reconstruct the secret are called qualified sets. Another family of the subsets of participants which do not obtain any information about the secret are called forbidden sets. The sets that are neither qualified nor forbidden can obtain partial information about the secret value. A secret sharing scheme is *perfect* if the forbidden sets coincide with the unqualified ones. In a *non-perfect secret sharing* scheme the length of some shares can be smaller than the length of the secret value. The connection between matroids and secret sharing was found in the Brickell-Davenport theorem [29] which states that every *ideal perfect secret sharing scheme* defines a matroid, which is uniquely determined by the access structure and the access structure is a port of this matroid. This result was generalized to non-perfect schemes by O. Farràs and C. Padró [69] where non perfect schemes were connected to quasi-matroids and generalized ports . A forbidden minor characterization for them was given by Seymour [138] which has found many applications in the study of secret sharing schemes and matroids. A forbidden minor characterization for generalized ports can advance the study of non-perfect schemes and their relations with matroids.

We make the following contributions :

1. A forbidden minor characterization for generalized matroid ports.

2. Connection between multipartite quasi-matroids, multipartite generalized ports and the associated integer polymatroid.

1.6 Combinatorics

1.6.1 A note on intervals in Hales-Jewett Theorem

One of the main results in conbinatorics and Ramsey theory is the Hales-Jewett Theorem. For $m, n \in \mathbb{N}$, let $[m]^n$ denote the set of all n-letter words with alphabets from $[m] = \{1, 2, \ldots, m\}$. For a word $w \in [m]^n$, $S \subseteq [n]$ and $i \in [m]$, $w(S, i)$ is obtained from w by replacing the jth letter with i for all $j \in S$. A *combinatorial line* in $[m]^n$ is defined as the set of words $\{w(S, 1), w(S, 2), \ldots, w(S, m)\}$ with the *wildcard set* $S \neq \phi$. The Hales-Jewett Theorem [86] says that for $m, r \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that any r-colouring of $[m]^n$ contains a monochromatic combinatorial line. In [53] the authors asked the following question:

• Do there exist $m \geq 4, r \geq 2$ and $c > 1$, $m, r, c \in \mathbb{N}$ such that there are r-colourings of $[m]^n$ containing no monochromatic combinatorial line whose wildcard set is the union of at most cr intervals ?

We answer this question in the positive : There is a 5-colouring of $[4]^n$ containing no combinatorial line whose wildcard set is the union of at most 25 intervals. If $gcd((m-1)(m-2), cr) = 1$, then the conjecture holds.

Chapter 2

Literature Survey

In this chapter, we discuss the existing literature on secret sharing, low complexity cryptography and secret sharing, different variants of secret sharing such dynamic secret sharing, evolving secret sharing etc. as mentioned in Chapter 1. We also look at the literature on AC^0 circuits, randomized decision tree complexity, quantum secret sharing and quantum query complexity. This chapter ends with the survey of some combinatorial results related to the intervals in the Hales-Jewett Theorem.

2.1 Classical Secret Sharing

To begin this survey, we refer the reader to [15] for an excellent survey on secret sharing schemes.

Secret sharing schemes were proposed independently by Shamir [139] and Blakley [20] in 1979. They proposed schemes where any k (or more) out of n participants are qualified to recover the secret with $1 < k \leq n$. The access structure is called a (k, n)-threshold access structure where a subset of size greater than or equal to the threshold value k is deemed to be qualified. Both schemes were fairly efficient in terms of the size of the shares and computational complexity. Later, Ito et al. [93] showed the possibility of constructing secret sharing schemes given any monotone (general) access structures. However, their generic constructions resulted in exponentially large share sizes. Later,

Karchmer et al. [96] provided scheme with share size is polynomial in the monotone span program complexity. A major objective in the area of secret sharing is to minimize the share size.

In classical secret sharing schemes it is generally assumed that the number of participants and the access structure is known in advance.

2.2 Dynamic Secret Sharing

Many secret sharing schemes have been proposed where the access structure changes over time. *Dynamic* secret sharing scheme allows, without reconstructing the shared secret, to add or delete shareholders, to renew the shares, and to modify the conditions for accessing the secret. This important primitive of redistributing the secret was initially considered by Chen et al. [44], Frankel et al. [72] and Desmedt-Jajodia [59].

To describe a dynamic secret sharing scheme more formally, let us consider two sets of participants $\mathcal{P}$ and $\mathcal{P}'$ containing n and n' many participants respectively. Let us suppose that each participant P_j in $\mathcal{P}$ has received a share s_j of the secret value s. $\Gamma_{\mathcal{P}}$ denote the access structure that specifies which subsets of $\mathcal{P}$ are *authorized* to recover the secret s from their shares. The *goal* of redistribution is that *without* the help of the original dealer, the participants in $\mathcal{P}'$ will receive the shares of s in accordance with a possibly different access structure $\Gamma_{\mathcal{P}'}$. In the protocol, the participants in $\mathcal{P}$ act like virtual dealers, while participants in $\mathcal{P}'$ are the ones who receive shares. A *notable* difference between evolving secret sharing and dynamic secret sharing is – in the former, dealer is present through out and he distributes new shares to joining parties.

Nojoumian-Stinson [124] proposed unconditionally secure share re-distribution schemes,

in absence of a dealer, based on a previously existing VSS protocol of Stinson-Wei [146]. In their construction, they have assumed less than one-fourth of participants behave dishonestly and also that the number of participants is fixed throughout. Their work was improved upon by the work of Desmedt-Morozov [60] who relaxed the proportion of dishonest participants to one-third of the total population and also allowed the number of participants to change.

A related primitive viz. sequential secret sharing (SQS) was introduced by Nojoumian-Stinson [125] as an application of dynamic threshold schemes. In this new primitive, different (but related) secrets with increasing thresholds are shared among a set of players who have different levels of authority. Subsequently, each subset of the players can only recover the secret in their own level. Finally, the master secret will be revealed if all the secrets in the higher levels are first recovered.

2.3 Evolving Secret Sharing

Komargodski et al. [99] introduced evolving secret sharing schemes where the secret holder a.k.a *dealer* does not know the number (or any upper bound) of participants that would participate in the protocol. Theoretically speaking, number of participants could be potentially infinite and the definition of the access structure may change as a function of time – a subset is declared to be qualified when the last participant who completes the set has arrived. The authors considered the scenario when players participate one at a time (in a sequential manner) and each player receives its share from the dealer. Main challenge in designing such a protocol is that the dealer cannot update the shares that he has already distributed. The authors [99] showed that for every evolving access

structure there exists a generic secret sharing scheme with 2^{t-1} as the share size of the t^{th} incoming participant. They also constructed (k, ∞)-threshold evolving secret sharing scheme for a fixed threshold value k with share size $(k-1)\log t + \mathcal{O}(\log\log t)$ of the t^{th} participant. Furthermore, they have provided an evolving 2-threshold scheme which is nearly optimal in the share size of the t^{th} participant viz. $logt + \mathcal{O}(loglogt)$.

Komargodski and Paskin-Cherniavsky [100] constructed evolving dynamic threshold schemes such that share size of the t^{th} participant is $\mathcal{O}(t^4 \log t)$ bits. Moreover, they used AMD codes to generically transform such evolving threshold schemes to *robust* schemes. Robustness of a secret sharing scheme means the correct secret is reconstructed even if some of the participants maliciously hand in tampered shares during the reconstruction process.

Later, Beimel and Othman [16] constructed evolving (a, b) ramp scheme with share size $\mathcal{O}(1)$ defined as follows : Let $0 < a < b < 1$. Any set of participants whose maximum participant is the i-th participant and contains at least ai participants can reconstruct the secret; however, we only require that any set such that all its prefixes are not a b-fraction of the participants should not get any information on the secret.

2.4 Secret Sharing in AC^0

The motivation to study secret sharing schemes that can be implemented by constant-depth circuits comes from two different sources. First, most well-known secret sharing schemes require computations that can not be implemented by constant-depth circuits (i.e. AC^0 circuits). For example, Shamir's scheme in [139] requires linear algebraic computations over finite field and hence cannot be computed in AC^0. Secondly, the visual

secret sharing schemes introduced by Naor and Shamir [120] require only computation of OR function which can be implemented by AC^0 circuit.

AC^0 is the complexity class which consists of all families of circuits having constant depth and polynomial size. The gates in those circuits are `NOT, AND, OR`, where `AND` gates and `OR` gates have unbounded fan-in. Integer addition and subtraction are computable in AC^0, but multiplication is not. It is also well known that calculating the parity of an input cannot be decided by any AC^0 circuits. For any circuit C, the size of C is denoted by `size(C)` and the depth of C is denoted by `depth(C)`. Recently, a lot of research [7, 8, 3, 13] have been done focusing on possibilities of obtaining cryptographic primitives in low complexity classes e.g. AC^0 or NC^1. We will later describe some primitives that are needed for our constructions.

A recent work by Bogdanov et al. [23] considers the question whether there exists secret sharing scheme such that both share generation algorithm and secret reconstruction algorithm are computable in AC^0.

They considered a variant of threshold secret sharing scheme, known as *ramp* schemes where any k participants learn nothing about the secret but when all n participants collaborate together, they are able to reconstruct the secret. The scheme is called ramp because unlike classical secret sharing scheme there is a gap between the privacy threshold viz. k and reconstructability threshold viz. n. Their construction connects the idea of *approximate degree* of a function with the privacy threshold of a secret sharing scheme. Existing literature on the approximate degree lower bounds gives several secret sharing schemes in AC^0. Their schemes however achieve large privacy threshold $k = \Omega(n)$ when the alphabet size is $2^{poly(n)}$ and achieve $k = \Omega(\sqrt{n})$ for binary alphabets. The work of Bogdanov et al. [23] was followed up by a work of Cheng et al. [47] who

achieved privacy threshold $k = \Omega(n)$ with binary alphabets by allowing negligible privacy error. They have also considered robustness of the schemes in presence of honest majority with privacy threshold $\Omega(n)$, privacy error $2^{-n^{\Omega(1)}}$ and reconstruction error $\frac{1}{poly(n)}$.

2.5 Secure Computation against moderately complex adversaries

The traditional cryptographic approach has been based on computational tasks which are easy for honest parties to perform and hard for the adversary. We have also seen a notion of moderately hard problems being used to attain certain security properties. Degwekar et al. [57] show how to construct certain cryptographic primitives in NC^1 [resp. AC^0] which are secure against all adversaries in NC^1 [resp. AC^0]. Ball et al. [13] present computational problems which are "moderately hard" on average. Continuing in this line Campanelli and Gennaro [36] prove that it is possible to construct secure computation primitives that are secure against *moderately complex* adversaries. They present definitions and constructions for the task of fully homomorphic encryption and Verifiable Computation in the fine-grained model. For possible applications of AC^0 secret sharing to secure broadcasting in presence of external adversaries we refer to Section 7.3 of [47].

2.6 Quantum Secret Sharing

Quantum secret sharing schemes[11, 51, 78, 111, 162] have been extensively studied using various tools, gates and techniques of quantum computation such as graph states, Bell states, GHZ- states, quantum teleportation[98, 106] etc. In a variant of secret sharing known as *threshold secret sharing*, the *access structure* (collection of all the qualified sets) consists of those sets whose cardinality is greater than a fixed number k. These schemes are denoted as (k, n)-schemes, where n is the total number of participants and k is the threshold. Quantum threshold secret schemes(QTSS) were studied in [14, 128, 131] and many more. The purpose of quantum secret sharing is twofold : 1. protecting classical information, 2. protecting quantum information. As noted by the authors in [131] sharing the quantum state is more difficult than sharing the classical information. So the quantum secret sharing schemes that share the quantum state are much fewer than the ones that share the classical information.

Some variants of secret sharing schemes allow to add or delete participants, to renew the shares, and to modify the conditions for accessing the secret. In the literature classical variants of these schemes are known by names such as dynamic, proactive, online secret sharing schemes etc. Quantum versions of these schemes were studied in [62, 92, 108, 129] and many more. Such constructions have certain limitations. For example, the dealer needs to know the number of new participants before-hand and the dealer also needs to know the set of qualified participants before handing out new shares.

Our construction uses repeatedly a (t, n)-quantum threhold secret sharing scheme [128, 131].

2.7 Quantum Hierarchical Secret Sharing

In this section, our focus is on hierarchical and multipartite access structures. Informally, multipartite access structures are those for which parties are grouped into different disjoint sets. The secret is shared among the participants and to reconstruct the secret, participants come from different groups with different properties to reconstruct the secret. Examples of these different properties can be 1. each group can have a different threshold, 2. participants need to arrive from at least a certain number of groups and many more. An important subclass of such secret sharing schemes is the *hierarchical secret sharing scheme.* Roughly, the groups of the participants are ordered and without the consent of participants higher in the hierarchy, the secret cannot be reconstructed. Hierarchical secret schemes are well-studied in the classical domain. For many of the hierarchical and multipartite access structures, ideal secret sharing schemes have been constructed.(Ideal secret sharing schemes are those where the share size equals the secret size). Also these access structures have been shown to be connected to a rich class of combinatorial structures known as matroids and polymatroids. For more on classical hierarchical secret sharing schemes we refer the reader to [17, 68, 97, 147, 148, 157, 161].

The quantum counterpart of hierarchical secret sharing schemes was studied in [153], where secret was distributed to three distant participants asymmetrically(participants have different powers to recover the sender's secret). In [154] the authors propose a multiparty asymmetric quantum secret sharing scheme with a multipartite-entanglement channel and classical communications. The scheme involves two grades of agents $G_1 = \{Bob_1, Bob_2, ..., Bob_m\}$ and $G_2 = \{Charlie_1, Charlie_2, ..., Charlie_n\}$. For get-

ting boss's (Alice's) secret state, one of Bobs needs the collaboration of the other Bobs and any one of Charlies, while one of Charlies needs the collaboration of all the other $m+n-1$ agents. A quantum hierarchical scheme was realized with a six-photon cluster state in [156]. Here a quantum secret is distributed to five distant agents who are divided into two grades. Two agents are in the upper grade and three agents are in the lower grade. An agent of the upper grade only needs the collaboration of two of the other four agents for getting the secret, while an agent of the lower grade needs the collaboration of all the other four agents. In [12] and [159] hierarchical quantum secret sharing schemes were proposed with a eight-cubit cluster state and two four-cubit cluster states respectively. Here also secret was distributed to two groups of parties with different hierarchies. Further studies of such hierarchical schemes were carried out in [117, 141] which added dynamism to the scheme where new parties could be added or deleted over time.

2.8 Randomized Decision Tree Complexity

A randomized decision tree A on n variables is a distribution over all deterministic decision tree algorithms on n variables. Given an input x, the algorithm first samples a deterministic tree $B \in_R A$ uniformly random, and then evaluates $B(x)$. The cost of a randomized algorithm A on input x, denoted in also by $C(A,x)$, is the expected number of input bits queried by A on x. Let P_f^δ be the set of randomized decision tree algorithms computing f with error at most δ. The two-sided bounded error randomized complexity of f with error $\delta \in [0,1/2)$ is

$$R_\delta(f) = min_{A \in P_f^\delta} max_{x \in \{0,1\}^n} C(A,x).$$

We write $R(f)$ for $R_0(f)$. Two functions for which exact values or bounds on the randomized complexity are known are the NAND_h and the *recursive majority* function, $3\mathsf{MAJ}_h$. NAND_h is the complete binary tree of height h with NAND gates, where the inputs are at the $n = 2^h$ leaves. Following is a well known result from [144].

Theorem 2.8.1. *(Snir [144]) $R(\mathsf{NAND}_h) \in O(n^c)$ where $c = \log_2(\frac{1+\sqrt{33}}{4}) \approx 0.753$.*

Let $\mathsf{MAJ}(x)$ be the Boolean majority function. The *recursive majority function* $3\mathsf{MAJ}_h$ is defined recursively on $n = 3^h$ variables, for every $h \geq 0$. For $h \geq 0$, let x be an input of length n and let $x^{(1)}, x^{(2)}, x^{(3)}$ be the first, second, and third $\frac{n}{3}$ variables of x. Then

$$3\mathsf{MAJ}_h(x) = \mathsf{MAJ}(3\mathsf{MAJ}_{h-1}(x^{(1)}), 3\mathsf{MAJ}_{h-1}(x^{(2)}), 3\mathsf{MAJ}_{h-1}(x^{(3)}))$$

In other words, $3\mathsf{MAJ}_h$ is defined by the *read-once* formula on the complete ternary tree T_h of height h in which every internal node is a majority gate. We identify the leaves of T_h from left to right with the integers $1, \ldots, 3^h$. We have a well known result from [112],

Theorem 2.8.2. *(Magniez et al. [112]) For all $\delta \in [0, \frac{1}{2}]$, we have $(\frac{1}{2} - \delta) \cdot 2.57143^h \leq R_\delta(3\mathsf{MAJ}_h) \leq (1.007) \cdot 2.64944^h$.*

2.9 Quantum Query Complexity

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, suppose $x = x_1 x_2 \ldots x_n \in \{0,1\}^n$ is an input of f (x_i denotes i-th bit). A *quantum query algorithm* for f computes $f(x)$, given queries to the bits of x. Quantum queries are made to an oracle which is defined

as $O_x |i, b\rangle = |i, b \oplus x_i\rangle$. A T-query quantum algorithms is a sequence of unitaries $U_T O_x U_{T-1} O_x ... O_x U_0$ where U_i's are fixed unitaries and O_x depends on x. The algorithm can be described as starting from a fixed state $|\phi_0\rangle$, it performs a sequence of unitaries as mentioned previously to obtain the final state $|\phi_x\rangle = U_T O_x U_{T-1} O_x ... O_x U_0 |\phi_0\rangle$. The state $|\phi_x\rangle$ is measured with a $0-1$ positive operator-valued measurement and the measurement result is regarded as the output of the algorithm. Let $m(x)$ denotes the measurement result of $|\phi_x\rangle$. If $\forall x$,

$$Pr[m(x) = f(x)] \geq 1 - \epsilon,$$

where $\epsilon < \frac{1}{2}$, then the quantum query algorithm is said to compute $f(x)$ with bounded error ϵ. If the error $\epsilon = 0$, then $f(x)$ is computed exactly. The *quantum query complexity* of a function f is the number of queries that an optimal quantum algorithm should make in the worst case to compute f. It is denoted by $Q_\epsilon(f)$ and in the exact setting it is denoted by $Q_E(f)$. For more on quantum query complexity we refer the reader to [5, 4, 6, 45, 105, 134].

2.10 Secret Sharing and Matroids

A secret sharing scheme is a method to distribute a secret value into shares among participants in such a way that only some qualified subsets of participants are able to recover the secret value from their shares. This family of subsets of participants are called qualified sets. Another family of the subsets of participants which do not obtain any information about the secret are called forbidden sets. The sets that are neither qualified nor forbidden can obtain partial information about the secret value. A

secret sharing scheme is perfect if the forbidden sets coincide with the unqualified ones. In a non-perfect secret sharing scheme the length of some shares can be smaller than the length of the secret value. The connection between matroids and secret sharing was found in the Brickell-Davenport theorem [29] which states that every ideal perfect secret sharing scheme defines a matroid, which is uniquely determined by the access structure and the access structure is a port of this matroid. This result was generalized to non-perfect schemes by O. Farràs and C. Padró [69] where non perfect schemes were connected to quasi-matroids and generalized ports. Matroid ports were introduced by Lehman [107] and a forbidden minor characterization for them was given by Seymour [138] which has found many applications in the study of secret sharing schemes and matroids. A forbidden minor characterization for generalized ports can advance the study of non-perfect schemes and their relations with matroids. The reader is referred to [15],[73], [114], [145] for surveys on secret sharing schemes, the connections between secret sharing schemes matroids and polymatroids, to [127] for a textbook on the theory of matroids and to [21], [18], [64], [65], [67], [103] for more on non-perfect secret sharing schemes and matroids.

2.11 Intervals in the Hales-Jewett Theorem

The Hales-Jewett Theorem [86] says that for $m, r \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that any r-colouring of $[m]^n$ contains a monochromatic combinatorial line. In [53], the authors proved the following : For any $n \in \mathbb{N}$ and any odd $r > 1$, there is an r-colouring of $[3]^n$ with no monochromatic combinatorial line whose wildcard set is the union of less than r intervals. In [53], the authors asked a question on the existence of monochromatic

combinatorial lines which we answer in chapter 9.

Chapter 3

Secret Sharing in AC^0

3.1 Introduction

All the classical secret sharing schemes assume that the number of participants as well as the access structure are fixed from the very beginning. An access structure is called an *evolving* access structure if the number of participants can grow without any bound and be potentially infinite with the possibility that the access sets are also changing over time. Existing classical methodology fails to provide a secret sharing scheme when the access structure is evolving. Some recent works have put forward secret sharing schemes for evolving access structures. In a recent development researchers have considered the problem of minimizing the computational complexity of cryptographic primitives and some recent positive results confirm the possibility of secret sharing with minimal computational complexity. More precisely, secret sharing with added randomness is possible with both share-generation algorithm and reconstruction algorithms are in the complexity class AC^0.

Hence the goal of this chapter is to bring these two together to construct AC^0 implementable secret sharing schemes which can add new participants over time.

3.1.1 Our Contribution

We construct dynamic secret sharing schemes and evolving schemes keeping the secret sharing and the reconstruction procedures in the complexity class AC^0. Our main idea is to use new share redistribution schemes based on good error-correcting codes [49, 47], and pseudorandom generators to give shares to the new nodes being added to a secret sharing scheme. The number of nodes is time-dependent and throughout the whole process (lifetime) of accommodating new nodes and share redistribution, secrecy of original data is maintained.

- Our first goal is to construct a dynamic AC^0 secret sharing scheme which can include new parties into the system even when the dealer is absent after generating shares of the old parties. To this end, using a secret redistribution scheme and by suitably modifying the scheme of [47], we construct a robust secret sharing scheme which can accommodate a bounded number of new parties. The advantage of redistribution here is that alphabet size need not be increased to accommodate new parties. The downside is, we can only add a bounded number of them.

- Next we construct a $(2, \infty)$ *evolving* secret sharing scheme in AC^0 where we can accommodate unbounded number of storage nodes but collective shares from any two nodes can reconstruct the secret. This theoretical result shows that threshold secret sharing with unbounded number of parties is possible with constant depth circuits and this can be of independent interest also. The generalization to an AC^0-implementable (k, ∞) evolving scheme is left as an open question.

3.2 Preliminaries

AC^0 complexity class. AC^0 is the complexity class which consists of all families of circuits having constant depth and polynomial size. The gates in those circuits are `NOT, AND, OR`, where `AND` gates and `OR` gates have unbounded fan-in. Integer addition and subtraction are computable in AC^0, but multiplication is not. It is also well known that calculating the parity of an input cannot be decided by any AC^0 circuits. For any circuit C, the size of C is denoted by `size(C)` and the depth of C is denoted by `depth(C)`. Recently, a lot of research [7, 8, 3, 13] have been done focusing on possibilities of obtaining cryptographic primitives in low complexity classes e.g. AC^0 or NC^1. We will later describe some primitives that are needed for our constructions.

Statistical Distance. The statistical distance between two random variables X and Y over Σ^n for some alphabet Σ, is $SD(X;Y)$ which is defined as follows,

$$SD(X;Y) = \frac{1}{2}\sum_{a\in\Sigma^n} |Pr[X=a] = Pr[Y=a]|.$$

Definition 5. *(Ramp Secret Sharing Scheme) A (k,l,n) ramp secret sharing scheme with $k < l \leq n$, on a set of n participants is such that any subset of participants of size greater than equal to l can recover the secret whereas, any subset of size less than k has no information about the secret.*

Definition 6. *(Evolving Secret Sharing Scheme [99]) Let $\mathcal{A} = \{\mathcal{A}_t\}_{t\in\mathbb{N}}$ be an evolving access structure. A secret sharing scheme $\mathcal{S}$ for $\mathcal{A}$ consists of a pair of algorithms $(SHARE, REC)$. SHARE is a probabilistic algorithm and REC is a deterministic algorithm which satisfy the following:*

1. *SHARE*$(s, \Pi_1^{(s)}, \Pi_2^{(s)}, \ldots, \Pi_{t-1}^{(s)})$ *gets as input a secret* s *from the domain of secrets* S *and the secret shares of participants* $1, 2, \ldots, t-1$ *and outputs the share of the* t^{th} *participant viz.* $\Pi_t^{(s)}$.

2. *(Correctness) For every secret* $s \in S$*, every* $t \in \mathbb{N}$ *and every qualified set* $B \in \mathcal{A}_t$*, it must hold that* $Pr[Rec(\{\Pi_i^{(s)}\}_{i \in B}, B) = s] = 1$.

3. *(Security) For every* $t \in \mathbb{N}$ *and every forbidden set* $B \notin \mathcal{A}_t$ *and for any two distinct secrets* $s_1 \neq s_2$ *in* S*, it must hold that the two distributions* $\{\Pi_i^{(s_1)}\}_{i \in B}$ *and* $\{\Pi_i^{(s_2)}\}_{i \in B}$ *are identical.*

Secret sharing scheme in AC^0 [23]. Let Σ denote set of alphabets. Two distributions μ and ν over Σ^n are called k-wise indistinguishable if for all subsets $S \subset [n]$ of size k, the projections $\mu|_S$ and $\nu|_S$ of μ and ν to the coordinates in S are identical. Thus, while sharing the secret bit 0 (resp. 1) if sampling is done using μ (resp. ν) then we see a direct connection to the fact that any k participants gain no information about the secret bit. However, if there is a function $f : \Sigma^n \to \{0, 1\}$ which can tell apart the distributions then f can be thought of as a reconstruction function. Of course, the gap between the privacy threshold k and the reconstructability threshold n makes the scheme a ramp scheme. The definition is as follows.

Definition 7. *(AC^0 Secret Sharing [23]) An (n, k, r) bit secret sharing scheme with alphabet Σ, reconstruction function $f : \Sigma^r \longrightarrow \{0, 1\}$ and reconstruction advantage α is a pair of k-wise indistinguishable distributions μ and ν over Σ^n such that for every subset S of size r we have $Pr[f(\mu|_S) = 1] - Pr[f(\nu|_S) = 1] \geq \alpha$.*

Minsky-Papert CNF function. The sharing function, Share, used in AC^0 constructions in the literature is based on the CNF function given by Minsky-Papert [116]. This

scheme can share one bit among n participants, with binary alphabet, privacy threshold $\Omega(n^{1/3})$ and perfect reconstruction.

Random Permutation. It is well-known that random permutation is in AC^0. For any $n \in \mathbb{N}$, a permutation over $[n]$ is defined to be a bijective function $\pi : [n] \to [n]$.

K-wise independent generators. A construction of K-wise independent generators based on unique neighbour expander graphs were proposed by Guruswami-Smith [82]. A set of n random variables, $X_1, \ldots, X_n$, is said to be k-wise independent(and uniform) if any k of them are independent(and uniformly distributed). For any $r, n, k \in \mathbb{N}$, a function $g : \{0,1\}^r \to \Sigma^n$ is a k-wise (uniform) independent generator, if for the uniform distribution U on $\{0,1\}^r$, the random variables $g(U) = \{Y_1, \ldots, Y_n\}$ are k-wise independent (and uniform).

Expander Graphs. A bipartite graph G with N left vertices, M right vertices is a (K, A) vertex expander if for all sets $S \subseteq [N]$ of at most K vertices, the neighborhood $N(S) = \{u | \exists v \in S : (u, v) \in E\}$ is of size at least $A\Delta|S|$.

3.2.1 Error-correcting codes

In coding-theoretic terms, the goal of secret sharing is to encode a secret S into a sequence $Y_1, \ldots, Y_n$ such that S can be recovered from the encoding and moreover for any $i_1, \ldots, i_t \in [n]$, the sequence $Y_{i_1}, \ldots, Y_{i_t}$ has the same distribution. In this light Shamir's scheme can be seen as following : a secret S is appended with t uniformly random and independent elements from a suitable finite field and the result is encoded using a Reed Solomon code of length n and dimension $t + 1$. Using coding-theoretic

properties one can prove that Shamir's scheme is in a sense robust. In this work we consider the optimal robust error-correcting codes/secret sharing scheme Cheng et al. [47]. The scheme of [47] is described in Section 3.3.1 and is AC^0 implementable.

Definition 8 (Robust secret sharing in AC^0 [47])**.** *For a secret x, if Y denotes the share string $Share(x)$ then for any adversary observing d shares and arbitrarily changing those values to transform the sharing string from Y to Y', the probability of correctly reconstructing the original secret is $Pr[Rec(Y') = x] \geq 1 - \eta$.*

3.3 Main results and technical details

In this section we present our main constructions with technical details. First we show how to construct a dynamic secret sharing scheme implementable in AC^0 where the existing shareholders accommodate new participant into the system and generate its share without the help of dealer. Our construction also achieves robustness thanks to the underlying basic scheme [47]. Second we give an AC^0 construction of evolving secret sharing scheme with reconstructability threshold 2 that can accommodate infinitely many parties. In this construction, we assume the dealer to be present to generate new shares. One of the basic differences between these constructions is that in dynamic secret sharing some or all of the old shares are modified whereas, in the evolving case no old shares are changed.

3.3.1 A dynamic robust secret sharing scheme in AC^0

Our construction of a dynamic secret sharing scheme is based on that of Cheng et al. [47]. We modify their scheme to accommodate new parties. For the ease of understand-

ing we first briefly describe their construction.

Overview of the construction of Cheng et al. [47]

For a short random seed R, it is shared using the one-in-a-box function [116] to get n shares with privacy threshold k_0. R and a k-wise independent generator are used to generate an n-bit string Y. To share a secret X, $Y \oplus X$ is computed. To reconstruct the secret, all the n parties are used to reconstruct R, compute Y and then compute X. This whole procedure can be computed in AC^0. To boost the privacy threshold and make the scheme robust the authors took the following steps:

(a) The parties are divided into blocks of size $O(\log^2 n)$.
(b) For each block a secret sharing scheme based on asymptotically "good" error-correcting codes is applied to obtain $O(\log^2 n)$ shares.
(c) These shares are further divided into $O(\log n)$ smaller blocks of size $O(\log n)$ each and a random permutation of these smaller blocks is applied. By increasing the alphabet size we can store each block together with its index permutation as one share.

The security of the scheme is argued in the following manner – if the adversary sees a constant fraction of the shares, since a random permutation is applied, the adversary learns each block with some constant probability. By using a Chernoff type bound combined with the fact that there are two levels of blocks, it can be ensured that the number of shares the adversary learns is below the privacy threshold of the larger block and thus the adversary actually learns nothing.

In **Fig. 3-1** we sketch the main steps of the share generation algorithm of [47]. The scheme uses random permutations, k-wise independent generators and asymptotically

good error correcting codes. The notations that were used (and we also use the same) in [47] are described below.

Notations. For any $n, k, m \in \mathbb{N}$ with $k, m \leq n$, alphabets Σ_0, Σ, let $(Share, Rec)$ be a k-out-of-n secret sharing scheme with share alphabet Σ, message alphabet Σ, message length m. Let $(Share_C, Rec_C)$ be an (n_C, k_C) secret sharing scheme from Lemma 3.13 of [47] with alphabet Σ, message length m_C, where $m_C = \delta_0 n_C$, $k_C = \delta_1 n_C$, $n_C = O(\log n)$ for some constants δ_0 and δ_1. For any constant $a \geq 1$, $\gamma \in (0, 1]$, [47] constructs a $(n_1 = O(n^a), k_1 = \Omega(n_1))$ secret sharing scheme $(Share_1, Rec_1)$ with share alphabet $\Sigma \times [n_1]$, message alphabet Σ, message length $m_1 = \Omega(n_1)$.

Dynamic construction. When a new party arrives, we take the following steps
(1) Add it in any of the larger blocks. Adding a new party in the larger block keeps the size of the block $O(\log^2 n)$.
(2) Generate share for the new party.
(3) Store the additional information e.g. the generation of the new party and to which block it is added multiples times.
(4) Divide the share into $O(\log n)$ blocks of size $O(\log n)$ each and proceed by applying the random permutation as before.

The share generation and reconstruction algorithms for the basic dynamic scheme accommodating just one new party are described in **Fig. 3-2** and **Fig. 3-3** respectively. Notice that we assume "centralized passive adversary model" where the parties under adversarial control follow the protocol but are interested in gaining information that they are not supposed to.

Discussions on the correctness & security. We sketch an overview of the correct-

1. The share generation algorithm is a function $Share_1 : \Sigma^{m_1} \to (\Sigma \times [n_1])^{n_1}$.

2. Let $\bar{n} = \Theta(n^{a-1})$ with large enough constant factor.

3. (*Independent generator step*) Let $g_\tau : \Sigma_0^{m\bar{n}} \to \Sigma^{m_1}$ be *l-wise independent generator* where $l = \Omega(\frac{m\bar{n}\log|\Sigma_0|}{\log|\Sigma|})^{1-\gamma}$.

4. For a secret $x \in \Sigma^{m_1}$, draw a string $r = (r_1, \ldots, r_{\bar{n}})$ *uniformly* from $\Sigma_0^{m\bar{n}}$.

5. Write $y = (y_s, y_g)$, where $y_s = (Share(r_1), \ldots, Share(r_n)) \in (\Sigma^n)^{\bar{n}}$ *and* $y_g = g_\tau(r) \oplus x \in \Sigma^{m_1}$.

6. Get $\hat{y}_s \in (\Sigma^{m_C})^{n_s}$ from y_s *by parsing* $y_{s,i}$ in blocks each having length m_C for every $i \in [\bar{n}]$, *where* $n_s = \lceil \frac{n}{m_C} \rceil \bar{n}$.

7. Get $\hat{y}_g \in (\Sigma^{m_C})^{n_g}$ *from* y_g *by parsing* y_g to blocks each having length m_C, *where* $n_g = \lceil \frac{m1}{m_C} \rceil$.

8. Compute

   $$(Share_C(\hat{y}_{s,1}), ..., Share_C(\hat{y}_{s,n_s}), Share_C(\hat{y}_{g,1}), ..., Share_C(\hat{y}_{g,n_g}))$$

   and parse it as $y1 = (y1_1, \ldots, y1_{n_1})$, *where* $n_1 = (n_s + n_g)n_C$.

9. (*Generate a random permutation*) $\pi : [n_1] \to [n_1]$ and apply it on $y1$ to get the desired output $\pi(y1) = Y$.

Figure 3-1: **The share generation algorithm of Cheng et al.** [47]

- **Share generation of an arriving party**

  1. Before the arrival of a new party, the old parties hold shares as generated in Fig. 3-1.
  2. Generate a share $S(T)$ for the new participant T by the following algorithm
     - Select two random parties from the old set of parties, say A, B
     - *Parse* shares of A and B as (A_1, A_2) and (B_1, B_2) respectively.
     - *New share* of A is (B_1, A_2) and the old share is deleted.
     - Share of B remains (B_1, B_2).
     - *Share of* the new party T is (A_1, B_1).
  3. This changes the shares string Y to Y_{temp} (due to the change to share of party A.)
  4. Concatenate $S(T)$ to Y_{temp} to get Y'_{temp}
  5. Store the relevant information multiple times.
  6. Apply a random permutation σ_T on the elements of the string Y'_{temp} to get the output $Y_T (= \sigma_T(Y'_{temp}))$.

Figure 3-2: **Share generation algorithm for the basic dynamic scheme in the passive adversary model.**

- **Reconstruction algorithm**

  1. *Compute the inverse permutation* $\sigma_T^{-1}(Y)$*to get* Y'_{temp}.
  2. *Remove* $S(T)$ *from* Y'_{temp} to get Y_{temp}.
  3. *Using* $S(T)$*, restore the original shares of the corresponding old parties and recover the string* Y.
  4. *Compute the inverse permutation* $\pi^{-1}(Y)$ *to get* $y1$.
  5. *Compute* Rec_C *on all the elements of* $y1$ *to get* y_s *and* y_g.
  6. *Apply Rec on every entry of* y_s *to get* r.
  7. *Output* $g_\tau(r) \oplus y_g$.

Figure 3-3: **Reconstruction algorithm for the basic dynamic scheme.**

ness and security properties here. Full details can be found in the Appendix. From the above algorithms it is easy to see the correctness – that is, $n_1 + 1$ shares together can reconstruct the secret. However, if n_1 out of the $n_1 + 1$ parties are chosen, they do not have complete information about the secret since the share of a party A has been changed. The dealer, at the beginning, can give an ordering to the parties and include the information multiple times $O(\log^2 n))$ in the shares. When a new party arrives the old shares are modified according to the order of the old parties. In the absence of dealer the parties modify their shares according to the order themselves. As a trade off we assume that a little storage is available to keep the information of the order of the shares. Combined with [47], all the remaining operations are AC^0 implementable and the scheme is robust. Hence we have constructed a robust dynamic AC^0 implementable secret sharing scheme. In this construction the share size is exponential. This robustness of this scheme follows from the usage of error-correcting codes. The size of the outer and the inner levels are $O(\log^2 n)$ and $O(\log n)$ respectively. To be

AC^0 implementable, size of the outer block must be $O(\log^2 n)$. Hence our scheme can accommodate upto $O(\log^2 n)$ new parties for each block.

We have the following theorem. A discussion on proof can be found in the Appendix at the end of this chapter.

Theorem 3.3.1. *For any $n, m \in \mathbb{N}$, $m \leq n$, any $\epsilon, \eta \in [0; 1]$ and any constant $a \geq 1$,$\alpha \in (0; 1]$, if there exists an explicit $(n' = O(n^a \log n); (1 - \alpha)n' = k')$ secret sharing scheme in AC^0 with share alphabet $\Sigma \times [n']$, message alphabet Σ_0, message length $\Omega(mn^{a-1})$, adaptive privacy error $O(n^{a-1})(\epsilon + 2^{-\Omega(k)}))$ and reconstruction error $O(n^{a-1}\eta)$, then, assuming a predefined order on the participants and a small storage to keep the information of the order of the participants, there exists an explicit $(n' + O(\log^3 n); (1 - \alpha)n')$ dynamic secret sharing scheme with adaptive privacy error $O(n^{a-1})(\epsilon + 2^{-\Omega(k)}))$ and reconstruction error $O(n^{a-1}\eta)$. The share and message alphabet and the message length of the new participants remain the same.*

Accommodating more parties

To accommodate more parties we may divide them into more equal sized blocks. $A = \{A_1, A_2, A_3, \ldots, A_f\}$ and $B = \{B_1, B_2, B_3, \ldots, B_f\}$. We can modify the shares of A as $\{B_1, A_2, A_3, B_4, \ldots\}$ and give the share of t as $\{A_1, B_2, B_3, \ldots\}$ or $\{A_1, B_2, A_3, \ldots\}$ and so on. Ours is a code based secret sharing scheme, so the number of blocks must be more than the distance of the code.

3.3.2 Evolving secret sharing in AC^0

We now give a construction which shows that an AC^0 secret sharing is possible for an evolving access structure where any two participants are qualified to reconstruct the secret whereas any one participant is unable to get any information about the secret bit. This result shows the possibility to include an unbounded number of participants in a secret sharing scheme where both the share generation algorithm and reconstruction algorithm are in AC^0.

Suppose the secret bit is $s \in \{0, 1\}$. Let $(\texttt{Share}_+, \texttt{Rec}_+)$ be a 2-out-of-2 threshold secret sharing scheme which can be obtained using the techniques of [23]. Applying this $(\texttt{Share}_+, \texttt{Rec}_+)$ algorithm multiple times, we show how the dealer prepares the shares for the participants.

Figure 3-5 gives a pictorial depiction of share generation process for $(2, \infty)$ access structure. Vertically shaded regions show the outputs of the basic (2-out-of-2) $\texttt{Share}_+$ algorithm run independently every time with the fixed secret bit s as input.

• The reconstruction algorithm is simple. When two participants come together they produce only the corresponding shares that connects them. Details are given in the following theorem.

Theorem 3.3.2. *There exists a $(2, \infty)$-secret sharing scheme implementable in AC^0 for which the share size of the t-th participant is linear in t.*

Proof. It is easy to see the share size of the t^{th} participant is linear in t. The proposed scheme runs the basic (2-out-of-2) AC^0 secret sharing scheme (independently) multiple times. So both the sharing and reconstruction phases can be implemented by AC^0

• **Share Generation**

1. The dealer first applies $\texttt{Share}_+$ algorithm on s and outputs (s_1^1, s_1^2).
   • Participant 1 receives s_1^1 as its share.
2. When participant 2 arrives, the dealer obtains $(s_2^2, s_2^3) \longleftarrow \texttt{Share}_+(\texttt{s})$ by an independent run.
   • Share of 2 is (s_1^2, s_2^2) and dealer stores s_2^3.
3. In general, for $t \geq 3$, share of the t-th participant consists of
   - the first $t-2$ entries of the share of $(t-1)^{th}$ participant
   - s_{t-1}^t [which is obtained as the second entry of the output of $\texttt{Share}_+(\texttt{s})$ (independently) run for the $(t-1)^{th}$ time to generate shares of participant $t-1$]
   - the random string s_t^t which is the first entry of $(s_t^t, s_t^{t+1}) \longleftarrow \texttt{Share}_+(\texttt{s})$ run for the t-th time.

Figure 3-4: **Share generation algorithm for $(2, \infty)$-evolving access structure.**

circuits.

To prove that any two participants can recover the secret let us suppose that participants i and j collaborate with each other. Without loss of generality, let $i > j$. We observe that participant j has s_j^j and it can collaborate with participant $j+1$ (who has s_j^{j+1}) to recover the secret. Recall that, $(s_j^j, s_j^{j+1}) \longleftarrow \texttt{Share}_+(\texttt{s})$ when run for the j-th time. Since share of participant i includes s_j^{j+1}, we have the proof.
The share generation algorithm ensures the secrecy of the scheme. □

Remark 1. *We observe that to improve the information rate of the scheme if we start with l bit secrets and assume the existence of a basic 2-out-of-2 AC^0 secret sharing scheme (for l bit secret) with negligible privacy error as in [47]. It is not very hard to see that the above construction gives a secret sharing sharing scheme with the same*

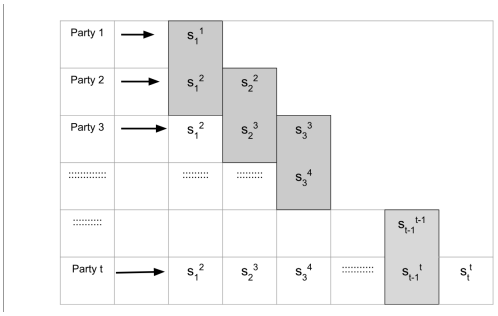

Figure 3-5: Step-construction of $(2, \infty)$ secret sharing scheme in AC^0.

privacy error as the basic one.

Example 1. *(Yet another example)*

Let us consider a star-graph based access structure where the internal vertex is fixed but the number of leaves changes/increases over time. A minimal qualified set is defined by two vertices which has an edge between them. More precisely, {`fixed internal node`, `any leaf`} *constitutes a minimal qualified set. Let* $(\texttt{Share}_+, \texttt{Rec}_+)$ *be a 2-out-of-2* AC^0 *implementable threshold secret sharing scheme. The dealer runs* $\texttt{Share}_+(\texttt{s})$ *(one time) to output* (s_1, s_2)*. Dealer assigns* s_1 *to the internal node and stores* s_2*. Whenever, a new leaf is added, the dealer assigns* s_2 *to the leaf.*

Discussion. At this point, it is not clear to the authors whether it is possible to construct other evolving threshold secret sharing schemes implementable in AC^0. Any possibility (or, impossibility) results is worth pursuing in future.

3.4 Comparison with existing schemes

Upon drawing comparisons with existing schemes in the literature, we observe the following : none of the existing dynamic schemes are AC^0 implementable. Another

advantage of our scheme is in the simplicity. New shares can be generated by simple manipulations of the share strings followed by random permutations. This avoids algebraic operations such as Lagrange's interpolations which are not AC^0 implementable. Note that depending on the choice of participants whose shares are modified, our scheme can be modified to make it into a hierarchical scheme and into many versions of multipartite schemes. The constructions are not difficult and we leave them for an expanded future version where we present AC^0 implementable constructions of various cases of multipartite secret sharing schemes which can accommodate new participants over time. The main drawback of the scheme (**Fig. 2** and **Fig. 3**) is that it can accommodate only a bounded number of new participants. We have attempted to overcome this drawback to construct an AC^0 implementable $(2, \infty)$ scheme. Ours is the first construction of an evolving secret sharing scheme which can accommodate potentially infinitely many participants over time. But our method cannot be generalized to construct AC^0 implementable scheme for general (k, ∞)-scheme for higher values k as there are more combinations of participants. Hence we leave that as an open problem.

Appendix A : Security Proof outlines of Theorem 3.3.1

Theorem 3.4.1. *$Share_1$ and Rec_1 can be computed by AC^0 circuits.*

Proof. We know that construction 1 can be done in AC^0. The extra functions that we are computing during adding a new participant are :

1. Generating the share $S(T)$ of the new participant. This can be done in AC^0 since

copying and concatenating string are AC^0-implementable operations.

2. Concatenating the share $S(T)$ to $y1$. This operation can be done in AC^0.

3. Applying a random permutation which is in AC^0.

For the reconstruction procedure, in our construction, the functions which we are computing other than those of [47] are

1. Inverse permutation σ_T^{-1}.

2. Restoring the original shares of the old participants.

3. Deleting the shares of some of the old participants.

Now the inverse permutation can be computed in AC^0. Restoring the share involves dividing a share into two halves and concatenating to the half of another share. Clearly this whole operation can be done in AC^0. The remaining deletion operation can be done in AC^0 too. Hence the $Share_1$ and Rec_1 functions can be computed in AC^0. □

Theorem 3.4.2. *Let the error during reconstruction of (Share; Rec) be η , then the error during reconstruction of $(Share_1, Rec_1)$ is $n' = \bar{n}\eta$.*

Proof. The reconstruction is done in two phases. First the shares of the new participants are used to restore the shares of the old participants. Next the old participants are used to reconstruct the secret. Although we need all the participants to reconstruct the secret, in the second phase it is the old participants who actually recover the secret. Hence our reconstruction error is essentially same as that of [47]. The proof is a simple application of the union bound in probability. □

Note : We stipulated that the adversary does not have any information regarding the order of the participants. So, from the adversary's point the old participants whose shares are modified when a new participant arrives is completely random and the share of the new participant is independent of the previous shares. Hence concatenating the share of the new participant does not affect the privacy of our scheme. Coupling this with the random permutation effectively results only in an increase in the length of the string. Hence our construction does not affect the privacy of the original scheme of **Fig.1**.

The overall effect is that the adversary only sees an increase in the number of repeated alphabets. Since the adversary sees only a constant fraction of shares, due to the repetitions and random permutations, it cannot infer any information about the secret. The details are given next.

In order to show privacy, the following Chernoff Bound is needed.

Negative Correlation.

Binary random variables $X_1, X_2, \ldots, X_n$ are said to be negatively correlated if for any subset I of $[n]$,

$$Pr[\wedge_{i\in I}(X_i = 1)] \leq \prod_{i\in I} Pr[X_i = 1]$$

and

$$Pr[\wedge_{i\in I}(X_i = 0)] \leq \prod_{i\in I} Pr[X_i = 0]$$

.

Theorem 3.4.3. *(Negative Correlation Chernoff Bound) Let $X_1, X_2, ..., X_n$ be random variables which are negatively correlated with $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}(X)$. Then*

1. *for any* $\delta \in (0,1)$, $Pr[X \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2}$ *and* $Pr[X \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}$.

2. *for any* $d \geq 6\mu$, $Pr[X \geq d] \leq 2^{-d}$.

Here we mention two lemmas regarding random permutations using which we can show the privacy of our scheme. For exact statements and proofs of these lemmas we refer the reader to Lemmas 3.7 and 3.8 of [47].

Lemma 3.4.4. [47] *Given* π *a random permutation of* $[n]$. *For any pair of sets* $S, W \subseteq [n]$, *let* $u = \frac{|W|}{n}|S|$. *The following items hold.*

1. *for any* $\delta \in (0,1)$, $Pr[|\pi(S) \cap W| \leq (1-\delta)\mu] \leq e^{-\delta^2\mu/2}$ *and* $Pr[|\pi(S) \cap W| \geq (1+\delta)\mu] \leq e^{-\delta^2\mu/3}$.

2. *for any* $d \geq 6\mu$, $Pr[|\pi(S) \cap W| \geq d] \leq 2^{-d}$.

Lemma 3.4.5. [47] *Let* π *be a random permutation of* $[n]$. *Let* $W \subseteq [n]$ *with* $|W| = \gamma n$. *Let* δ *be constant* $\delta \in (0,1)$. *Let* $t, l \in \mathbb{N}^+$ *such that* $tl \leq \frac{0.96}{1+0.96}\gamma n$. *Let* S *be a collection of subsets* $\{S_1, ..., S_l\}$ *such that for each* $i \in [l]$, *the sets* $S_i \subseteq [n]$ *are disjoint and* $|S_i| = t$. *Finally, let* X_i *be the indicator random variable such that* $X_i = 1$ *is the event* $|\pi(S_i) \cap Wj| \geq (1+\delta)\gamma t$. *Taking* $X = \sum_{i \in [l]} X_i$, *we have for any* $d \geq 0$, $Pr[X \geq d] \leq e^{-2d + (e^2-1)e^{-\omega(\gamma t)l}}$.

Using the above lemmas one can show privacy of the secret sharing scheme as follows.

Lemma 3.4.6. [47] *Let* Σ *be a set of alphabets and let* $n, k \in \mathbb{N}$ *with* $k \leq n$. *Given a distribution* $X = (X_1, ..., X_n)$ *over* Σ^n, *let* Y *be the distribution obtained by the action of* π^{-1} *on* X *where* $\pi : [n] \to [n]$ *is a random permutation. If an adaptive adversary observes a set of coordinates* W *with* $|W| = k$ *then* Y_W *is the same distribution* $Y_{[k]}$.

Note : This lemma essentially says that due to the random permutation the adversary observing a constant fraction of the secret cannot learn anything about the secret.

Utilizing the above-mentioned lemmas we have the following theorem estimating the parameters in our case.

Theorem 3.4.7. *Let $n, m \in \mathbb{N}$, with $m \leq n$, $\epsilon, \eta \in [0;1]$ and constant $a \geq 1$,$\alpha \in (0;1]$. Suppose we have an explicit $(n' = O(n^a logn); (1-\alpha)n')$ secret sharing scheme computable in AC^0 with share alphabet $\Sigma \times [n']$, message alphabet Σ_0, message length $\Omega(mn^{a-1})$, adaptive privacy error $O(n^{a-1})(\epsilon+2^{-\Omega(k)}))$ and reconstruction error $O(n^{a-1}\eta)$, then, assuming a predefined order on the participants and a small storage to keep the information of the order of the participants, an explicit $(n'+O(log^3 n); (1-\alpha)n')$ dynamic secret sharing scheme with privacy error $O(n^{a-1})(\epsilon + 2^{-\Omega(k)}))$ (adaptive) and error of reconstruction $O(n^{a-1}\eta)$ can be constructed.*

Chapter 4

Perpetual Secret Sharing Scheme from Fractional Cascading

In chapter 3 we constructed an AC^0 implementable dynamic secret sharing scheme and another AC^0 implementable $(2, \infty)$ evolving scheme. The constructed evolving scheme cannot be generalized to a general (k, ∞) scheme and the dynamic scheme can only accommodate a bounded number of new parties. In this regard, we introduce the idea of *perpetual* secret sharing [40] – a combination of evolving and dynamic secret sharing in which some (not all) old shares are changed to generate new shares and an infinite number of parties can be accommodated in the system. Two important features of our construction are – the dealer does not have to use a huge memory (unlike evolving schemes which use exponential amount of memory) and also that our scheme is implementable by AC^0 circuits. AC^0 is the lowest complexity class in which share distribution and secret reconstruction is possible. This makes our scheme more implementable. To this end we use a dynamic data structure technique known as fractional cascading [41, 42] to accommodate, distribute and renew shares among parties. We note that such data structures are not only used to accommodate new parties, but this data structure takes an active part in how the shares of old parties are redistributed and given to new parties and are helpful in keeping the complexity of the computations very low. In the concluding section we indicate how to use other dynamic data structures

such as Tango trees etc. to produce different access structures which can accommodate new parties. Such dynamic versions of data structures also help to delete parties, push a party up or down a hierarchical order which are in turn helpful to construct dynamic or evolving versions of hierarchical secret sharing schemes, multipartite secret sharing schemes and more. Hence we can achieve long term confidentiality of such dynamic schemes with low computation resources. In order to achieve this we also introduce a new formal definition for secret redistribution.

The main contribution of this chapter can be summarized as follows.

1. We construct a secret sharing scheme which can accommodate an unspecified and unbounded number of parties keeping the share and the reconstruction process in the complexity class AC^0.

2. The construction utilizes fractional cascading to organize and pre-process the parties. The access structure is determined by this data structure.

4.1 Preliminaries

We only mention the most relevant definitions and refer the reader to appropriate references for the commonly used definitions. For the formal definition of secret sharing we refer the reader to [139, 20, 93]. For evolving secret sharing schemes we refer the reader to the works of Komargodski et al. [99, 100, 16]. We begin with our formal definition for secret redistribution.

Secret Redistribution. We introduce the following definition.

Definition 9. *An (n, k)-redistribution scheme consists of two groups of parties of sizes n and k, $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_k$ respectively. The parties $P_1, \ldots, P_n$ have predefined shares as per some secret sharing scheme. A redistribution scheme modifies the shares of $P_1, \ldots, P_n$ to compute $n + k$ new shares such that:*

1. *Original shares of $P_1, \ldots, P_n$ are deleted.*
2. *New $n + k$ shares are distributed among all the $n + k$ parties $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_k$.*
3. *All parties $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_k$ combining can reconstruct the original shares of $P_1, \ldots, P_n$.*
4. *parties $P_1, \ldots, P_n$ cannot use original shares before the reconstruction stage.*
5. *parties $P_1, \ldots, P_n$ themselves cannot obtain original shares from new shares.*

Notation : (n, k)-redistribution scheme is denoted by the pair $(\mathsf{Redist}_{GEN}^{(n,k)}, \mathsf{Redist}_{REC}^{(n,k)})$. The algorithm $\mathsf{Redist}_{GEN}^{(n,k)}$ generates $n + k$ new shares from n old shares and $\mathsf{Redist}_{REC}^{(n,k)}$ combines new shares to output old shares.

Redistribution schemes are constructed using pseudo-random generators or by coding theoretic techniques combined with random permutations. More details are given in section 4.3.

Fractional Cascading. Fractional cascading was introduced by Chazelle and Guibas [41]. It is a data structure technique used for iterative searching in a collection of k ordered lists or *catalogs*. Suppose we have k ordered lists $L_1, ..., L_k$ such that $|L_i| \leq n \; \forall$ $1 \leq i \leq k$. The next step is to modify these lists as follows : Denote the modified lists as

$\bar{L_1}, ..., \bar{L_k}$. Set $\bar{L_k} = L_k$ and for each i, $1 \leq i \leq k-1$, $\bar{L_i}$ is obtained by merging (adding elements at proper positions) L_i with every other element of $\bar{L_{i+1}}$. The next step is to add *connections (bridges/pointers)* between these modified lists. For $1 \leq i \leq k-1$, if an element in a list $\bar{L_i}$ came from L_i, add pointers to the two nearest elements in $\bar{L_{i+1}}$. In addition to this, add connections between the merged elements between $\bar{L_i}$ and $\bar{L_{i+1}}$. These connections are helpful to find the ranges of the location of the required element to be located. To search an element in the collection of catalogs, first a binary search is done in the first modified list $\bar{L_1}$. If the element is not found, the connections are used to locate the element in the next modified list $\bar{L_2}$ and so on. The advantage of fractional cascading is that the operation takes $O(\log n + k)$ time and $O(n)$ space. For our purpose, in addition to efficient iterative searching, this structure along with the connections gives us a platform for redistributing secret. As shown in figure 1, secret is redistributed along the connections. We refer the reader to [42] for more on fractional cascading and its applications.

Perpetual Secret Sharing

Definition 10. *A perpetual secret sharing scheme on an evolving access structure (an access structure which adds new participants over time) is a secret sharing scheme with the following conditions :*

- *The access structure evolves according to a dynamic data structure. This means that at any time if a set of participants is connected to each other through connections determined by the underlying data structure and the set of participants is maximal(no other participant is connected to any of these participants), we shall call such a set of participants a quasi-qualified set of participants who should be*

able to reconstruct a predetermined fraction of the secret. The quasi-qualified sets can combine their partial fractions of the secret to reconstruct the original secret.

Remark 2. *In our construction (see Fig 1.) we shall see that, a secret is divided among participants of L_1 and the secret is redistributed among the remaining participants in the catalogs L_2, L_3 and L_4 along the connections. So the quasi-qualified sets are $\{24, 23, 13, 11\}$, $\{64, 25, 26\}$, $\{65\}$, $\{80, 68, 44, 62, 66, 35, 46\}$ and $\{93, 90, 87, 79, 81\}$. Partial secrets are reconstructed by each of these subsets and finally the participants 24, 64, 65, 80 and 93 combine to reconstruct the original secret. Now as new participants arrive, connections are updated and these quasi-qualified sets change/evolve over time.*

Remark 3. *In secret-sharing literature, qualified sets sets are those subsets of participants who can reconstruct the secret. So, quasi-qualified subsets are certain specialized subsets of participants who can reconstruct a predetermined part of the secret.*

4.2 Technical Details : Fractional Cascading based Dynamic Secret Sharing

Pre-processing. We denote participants by positive integers and call them the *weights* of these participants. Participants do not arrive in any order of their weights but they are stored in the lists in increasing order. When a new participant arrives, it is added to a suitable list and the share of certain (bounded number of) parties are changed. Initially we set that each list can accommodate at most n participants.

Overview of our idea. Let us suppose that at any instance we have k ordered catalogs of participants in increasing order where each participant is denoted by a natural

number called *weight*. The lists of parties are $L_1, ..., L_k$. The first party x_1^1 in the list L_1 is *connected* to all those parties in L_2 whose weights are less than or equal to x_1^1. The second party x_1^2 in L_1 is connected to all those parties in L_2 which have not been previously connected and whose weights are less than or equal to x_1^2 and so on. We repeat the previous step for parties in L_2 and L_3 and so on. Note that initially, there can be parties which are not connected to any other party. Each party along with the connections in the list L_1 is the root of a tree .

Let us suppose at this stage a new party with *weight* q arrives. If $q < max(L_k)$, insert q in L_k maintaining the order and update the connections between L_{k-1} and L_k. If $q > max(L_k)$ and $q < max(L_{k-1})$, insert q in L_{k-1} keeping the order and update the connections between L_{k-2}, L_{k-1} and L_k. If $q > max(L_{k-1})$ go to L_{k-2} and so on. If $q > max(L_1)$, add q to the end of L_1.

In this structure, the lists will be called generations interchangeably keeping parity with similar constructions in the literature [99, 100, 16]. They are based on range of the weights of the parties. After certain time and adding participants when all the generations are exhausted, the new party is added to L_1. The sizes / ranges of the previous generations are increased suitably and the process is repeated. As per requirement one may also create a new generation and proceed.

Sharing secret : To share a secret S, first the secret is distributed among all the parties of L_1. As mentioned above, one party can be attached only to bounded many parties in the next list. Hence each of the shares in L_1 are distributed to bounded many parties in L_2 and so on along the connections. When new parties arrive, share

is redistributed along the connections. parties in one list combine to reconstruct the share of the previous list. Share sizes do not increase drastically due the nature of the structure and pre-processing. The process is formalized in Algorithm 1.

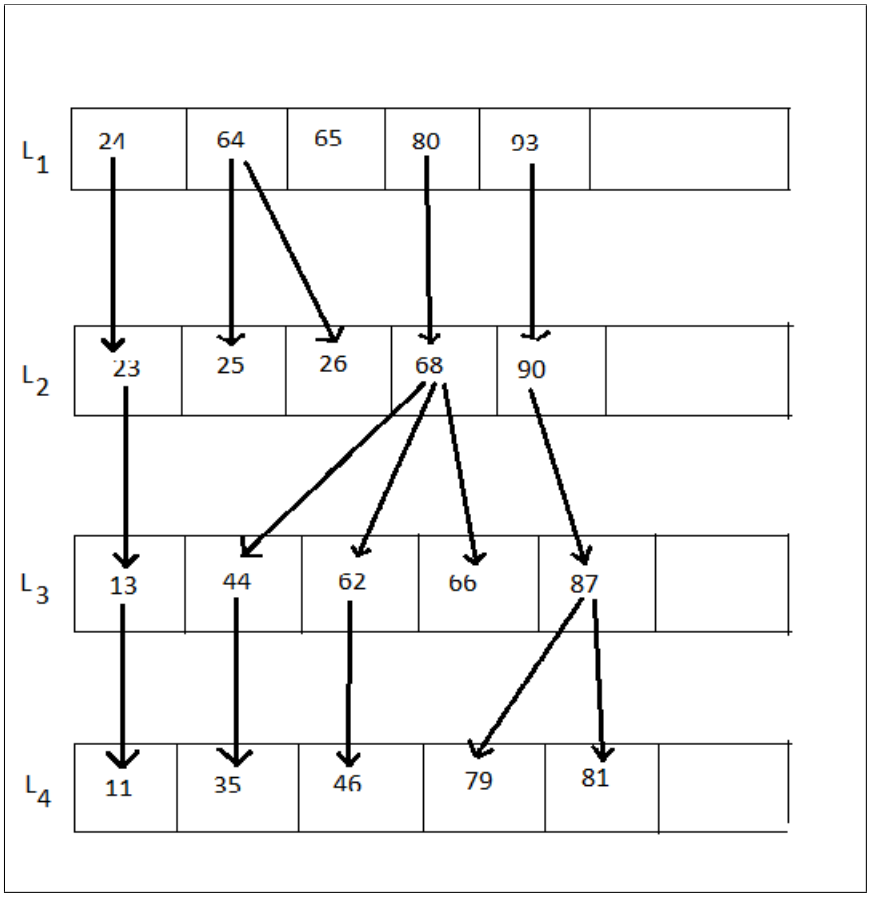

Figure 4-1: Connecting parties among ordered lists.

Notation: Given k lists $L_1, L_2, \ldots, L_k$ each of size at most n. Each list is filled with parties denoted by their weight in increasing order. Here $size(L_i)$ denotes the number of parties in the list L_i. Elements of the lists are denoted by $L_i[.]$. To store the connections, for each party maintain lists $C_{(p,q)}[.]$, where p, q denotes the list and the position in the list respectively.

Algorithm 1 Combining fractional cascading and secret sharing

procedure CREATING INITIAL CONNECTIONS
Initialize $i = 1$, $j = 1$, $g = 1$.
While $i \leq k - 1$ do steps 4 to 8.
While $j \leq size(L_i)$ do steps 5 to 7.
While $g \leq size(L_{i+1})$ do step 6.
If $L_{i+1}[g] \leq L_i[j]$ then add g to the list $C_{(i,j)}$ else increase g by 1.
Increase j by 1.
Increase i by 1.

procedure ACCOMMODATING NEW PARTIES
New party denoted by its weight q is to be included.
If $q < max(L_k)$, insert q in a proper position in L_k and update the connections between L_k and L_{k-1}. Else check in L_{k-1}.
If $q < max(L_{k-1})$, insert q in a proper position in L_{k-1} and update the connections between L_{k-2} and L_{k-1}. Create new connections between L_{k-1} and L_k. Else check in L_{k-2} and so on.
Repeat step with L_{k-j} until we reach L_1.
Insert/Add q in a proper position in L_1.

procedure SHARING SECRET TO NEW PARTY
Suppose party with weight q is added to the list L_j.
If $j = 1$ then,

1. Assign n as the current number of old parties in L_1.
2. Run $\mathsf{Redist}_{GEN}^{(n,1)}$ to redistribute the shares of the old parties in L_1 and give share to the new party.
3. Modify shares of the next lists $L_2, ..., L_k$ via proper redistribution schemes.

If $j > 1$ then,

1. Locate the parent of q in the list L_{j-1} and the other parties in L_j connected to the same parent. Count all such old parties along with the parent and assign the count as C.
2. Run $\mathsf{Redist}_{GEN}^{(C,1)}$ to redistribute the shares of the old parties and the parent in L_j and L_{j-1} respectively and give share to the new party q.
3. Modify shares along connected parties in the lists $L_{j+1}, \ldots, L_k$ via proper redistribution schemes.

Remark 4. *Following this procedure we must delete the duplicate connections. This may happen because from the algorithm an element in a list can be connected from two distinct elements in the previous list. In such a case the connection from the party with greater denomination is deleted.*

Remark 5. *It is clear from Fig. 4-1 that each of the quasi-qualified sets form a tree and hence the whole structure becomes a forest. One more reason to consider such a dynamic data structure / forest based construction is that different generations may have different sizes. Also in many practical scenarios, one may need to add new parties to an earlier generation as per hierarchical requirements. Our construction can support such scenarios.*

Share distribution.

1. Let us suppose that initially there are n many elements in the list L_1. The dealer can run an (n, n) scheme to generate n shares for L_1. To keep the complexity low, an AC^0-scheme is used (details later).

2. If a party in L_1 is connected to t parties in L_2, use a $(1, t)$-redistribution scheme, to redistribute shares among the parties in L_2. Repeat the procedure for parties in L_2 and their connections in L_3 and so on.

3. When new parties arrive, the connections are updated and shares are redistributed as per the updated connections in the lists.

Secret Reconstruction. Parties in the last list L_k combine their share to reconstruct the shares of the previous list L_{k-1} and similarly the shares of the parties of the lists

L_{k-2}, L_{k-3} and so on are recovered until we reach the list L_1. The parties of the list L_1 combine their shares to reconstruct the secret as per the reconstruction procedure of the (n, n) secret sharing scheme.

Theorem 4.2.1. *Quasi-qualified sets combining can reconstruct the secret. No other subset of participants have any information about the secret.*

Proof. In this scheme, whenever a new list is created, the shares of the parties in the previous list gets modified. The parties in the last generation/list combine to form the partial shares of the parties in the previous generation who in turn combine to reconstruct the partial shares of the parties in the generation one level above and so on. Hence the quasi-qualified sets combining can reconstruct the secret.(For correctness of the secret redistribution scheme see section 4). Now consider any proper subset of participants which is not a quasi-qualified set. Therefore there is at least one participant in some list/generation which is not in this subset. By the reconstruction procedure, the partial secret of that generation cannot be recovered and hence the original secret cannot be recovered. For the proof that this subset has no information about the secret see section 4.3.3. □

Remark 6. *From the above proof note that no participant is redundant to the system. Hence essentially we have constructed an n_t-out-of-n_t secret sharing scheme where n_t is the total number of participants at time t and the quasi-qualified sets evolve as per the underlying data structure. Hence we have constructed a perpetual secret sharing scheme from fractional cascading.*

Remark 7. *Once the number of participants become very large we have the added advantage of efficient searching. Using dynamic variants of fractional cascading we can*

delete participants from this scheme also.

4.3 Share distribution, redistribution schemes and share size

In this section we the share distribution and redistribution schemes.

4.3.1 Share distribution and redistribution schemes

(n, n)-scheme. To generate shares for the list L_1, we can use Shamir's (n, n)-scheme. But Shamir's scheme is not AC^0 computable. In such a case we can use the AC^0 scheme of [47]. While Shamir's scheme is ideal and it implies less share size, the scheme of [47] has exponential share size. The privacy threshold is $\Omega(n)$.

4.3.2 Redistribution Schemes

We first look at the definition of a random partition.

Definition 11. *A random partition of a string into p parts is a random permutation of the elements of the string followed by partitioning the string into p equal parts.*

As an example let us suppose that the share S_1^1 has to be redistributed into shares S_1^{11} and S_1^{12}. There are two ways to do this. Firstly encode S_1^1 using an (n, k)-code where the operations can be done in AC^0 [47]. Now partition the coded string into three equal parts using a random partition to generate three shares S_1', S_1^{11} and S_1^{12}.

Secondly one may use a pseudorandom generator instead of codes to extend the message and generate the shares S_1^{11} and S_1^{12}. The procedure for the $(1,2)$-redistribution scheme is formalized below.

Algorithm 2 Redistribution of secret shares

procedure SHARE REDISTRIBUTION $(\mathsf{Redist}_{GEN}^{(1,2)})$
    Encode(S) using a coding scheme to generate $Enc(S)$.
    Use random partition to split $Enc(S)$ to S_1, S_2 and S_3.
    Keep S_1 for old party whose share is being modified.
    Distribute S_2 and S_3 to two new parties.
procedure SHARE RECONSTRUCTION $(\mathsf{Redist}_{REC}^{(1,2)})$
    Input: S_1, S_2, S_3
    Concatenate S_1, S_2, S_3 to get S^1.
    Apply inverse permutation on S^1 to get $Enc(S)$.
    Output: $Dec(Enc(S)) \longrightarrow S$.

The same procedure can be generalized to case of a $(1,k)$-redistribution scheme. If we use pseudorandom generators we use the following modified algorithm 3.

Algorithm 3 Redistribution using pseudorandom generators

procedure SHARE REDISTRIBUTION $(\mathsf{Redist}_{GEN}^{(1,k)})$
    Stretch S using a pseudorandom generator to get $\bar{S}$.
    Use a random permutation to permute the elements of $\bar{S}$.
    Split $\bar{S}$ into $k+1$ equal parts, $S_1, \ldots, S_{k+1}$.
    Distribute $S_1, \ldots, S_{k+1}$.

4.3.3 Proof outline for redistribution schemes

Theorem 4.3.1. *Algorithm 2 and 3 satisfy the properties 1-5 of redistribution schemes.*

Proof. We note that these redistribution schemes of algorithms 2 and 3 follow Definition 9. Clearly the properties $1-4$ of Definition 9 are satisfied. To see that property 5 is

satisfied we recall the redistribution process. First the original shares of some particular parties are concatenated to get a single string. This string is encoded via a coding scheme to get an encoded string. Following this a random partition is applied. This is the crucial step. A random partition is a random permutation of the elements of the string followed by the division of the string into some equal parts. Due to the random permutation an old party cannot distinguish between an old share and new share. Even if a constant fraction of the string is observed, no information can be obtained from the new string. For more details, we refer the reader to the Appendix at the end of this chapter. □

Remark 8. *Since there are only finitely many permutations of a string of N elements, we can order these permutations and embed the order of the permutation used in the string for the parties to use during the reconstruction process. This adds a linear overhead to the share size. Some other information we need to store are the size of the partitions which adds a constant overhead to the share size.*

Proof. (Continuation of proof of **Theorem 4.2.1.**) Suppose in a subset a participant from a list is missing. Consider the participants in that list which have a common parent participant in the previous list. For example in Fig. 1, 68 is the parent of 44, 62 and 66. By the construction of redistribution schemes, with the corresponding partition missing, the remaining participants does not know the exact positions of the remaining permuted elements and hence have no information about the complete string. In the next higher list, by the similar argument, the participants have no information about the partial secrets. Hence the whole effect is the any set which is not quasi-qualified has no information about the secret and hence has no information about the original

secret. □

4.4 Complexity analysis

Theorem 4.4.1. *The Share distribution, Redistribution and secret reconstruction can be done by AC^0 circuits.*

Proof. The arriving parties are included in k lists. When a new party arrives , the total number of connections that needs to be updated is k, since there are k-many lists. Also the number of partitions into which the share has to be redistributed is bounded by a constant. This constant depends on the initial values in the k lists. Hence we can assume that there is a constant C, such that the total number of partitions needed to redistribute a share is bounded above by C. To redistribute a share we use an (cn, n)-error correcting code to encode the share[47] or use pseudorandom generators[83]. This process can be done in AC^0. Next we use a random permutation to permute the elements of the extended string and then split the string into at most C parts. Both these processes and hence the random partitions[115, 84, 150] can be achieved by AC^0 circuits.

During reconstruction, the parties of one generation(list) combine to reconstruct the share of the previous generation(list). This is done by concatenating the shares, applying the inverse permutation and decoding the share. These are well-known facts that all these processes can be done in AC^0. □

4.5 Comparison with existing schemes

The two main advantages of our scheme over the evolving schemes of [99, 100] are

1. Our construction uses negligible amount of memory in comparison with both the evolving schemes in literature.

2. Our construction is AC^0 implementable while none of the existing dynamic or evolving schemes are AC^0 implementable.

3. Also instead of using fractional cascading we can use dynamic data structure techniques such as tango trees and others to implement dynamic or evolving versions of several other secret sharing schemes such as hierarchical secret sharing, multipartite secret sharing and many more. So our construction is more efficient and flexible than the existing ones. Also the use of data structures allow us to search for parties and delete parties and these are particularly useful when the number of parties become very large. Note that we do not increase the size of the underlying field and hence the share size does not increase drastically as new parties arrive.

Appendix -B

In this section we include the proof sketch of the fact that after applying a random permutation, the original share cannot be distinguished from the permuted share.

Negative Correlation

Binary random variables $X_1, X_2, ..., Xn$ are negative correlated if $\forall I \in [n]$,

$$Pr[\wedge_{i \in I}(X_i = 1)] \leq \prod_{i \in I} Pr[X_i = 1]$$

and

$$Pr[\wedge_{i \in I}(X_i = 0)] \leq \prod_{i \in I} Pr[X_i = 0]$$

.

Theorem 4.5.1. *(Negative Correlation Chernoff Bound). Let $X_1, X_2, ..., X_n$ be negatively correlated random variables with $X = \sum_{i=1}^{n} X_i$, $\mu = \mathbb{E}(X)$. Then*

- *for any $\delta \in (0, 1)$, $Pr[X \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu/2}$ and $Pr[X \geq (1 + \delta)\mu] \leq e^{-\delta^2 \mu/3}$.*
- *for any $d \geq 6\mu$, $Pr[X \geq d] \leq 2^{-d}$.*

Here we mention two lemmas regarding random permutations using which we can show the privacy of our scheme. For proofs of these lemmas we refer to Lemmas 3.7 and 3.8 of [47].

Lemma 4.5.2. *Let $\pi : [n] \to [n]$ be a random permutation. For any set $S, W \subseteq [n]$, let $u = \frac{|W|}{n}|S|$. Then the following holds.*

- *for any $\delta \in (0, 1)$, $Pr[|\pi(S) \cap W| \leq (1 - \delta)\mu] \leq e^{-\delta^2 \mu/2}$ and $Pr[|\pi(S) \cap W| \geq (1 + \delta)\mu] \leq e^{-\delta^2 \mu/3}$.*
- *for any $d \geq 6\mu$, $Pr[|\pi(S) \cap W| \geq d] \leq 2^{-d}$.*

Lemma 4.5.3. *Let* $: [n] \to [n]$ *be a random permutation. For any* $W \subseteq [n]$ *with* $|W| = \gamma n$*, any constant* $\delta \in (0,1)$*, any* $t, l \in \mathbb{N}^+$ *such that* $tl \leq \frac{0.96}{1+0.96}\gamma n$ *any* $S = S_1, ..., S_l$ *such that* $\forall i \in [l]$*,* $S_i \subseteq [n]$ *are disjoint sets and* $|S_i| = t$*, let* X_i *be the indicator such that* $X_i = 1$ *is the event* $|\pi(S_i) \cap Wj| \geq (1+\delta)\gamma t$*. Let* $X = \sum_{i \in [l]} X_i$*. Then for any* $d \geq 0$*,* $Pr[X \geq d] \leq e^{-2d+(e^2-1)e^{-\omega(\gamma t)l}}$.

Using the above lemmas one can show privacy of the secret sharing scheme as follows.

Lemma 4.5.4. *For any alphabet* Σ*, any* $n, k \in \mathbb{N}$ *with* $k \leq n$*, for any distribution* $X = (X_1, ..., X_n)$ *over* Σ^n*, let* $Y = ((X_{\pi^{-1}(1)} \circ \pi^{-1}(1)), ..., (X_{\pi^{-1}(n)} \circ \pi^{-1}(n)))$ *where* π *is a random permutation over* $[n] \to [n]$*. For any adaptive observation* W *with* $|W| = k$*,* Y_W *is the same distribution as* $Y_{[k]}$.

For the proof of this lemma we refer the reader to Lemma 3.10 of [47]. This lemma essentially says that due to the random permutation the adversary observing a constant fraction of the secret cannot learn anything about the secret.

Chapter 5

A Quantum Evolving Secret Sharing Scheme

In this chapter we construct an *evolving quantum secret sharing scheme*(EQSSS) which can share and protect a secret quantum state. While we use well-studied methods of [99, 100] to generate share for the new participant, to the best of knowledge of the author, the quantum version of an evolving secret sharing scheme has not been considered prior to our work [37].

Our construction uses repeatedly a (t, n)-QTSS scheme [128, 131]. We shall omit the exact details of these constructions. Thus the contribution can be summarized as follows.

• **Our Contribution**

1 : Construction of an evolving quantum secret sharing scheme (EQSSS) which shares and protects a secret quantum state.

2 : Discussion of some possible ways to reduce the usage of quantum memory.

This chapter is more algorithmic in flavor but it poses some implementation challenges which can of independent interest both to physicists and computer scientists alike. We discuss these issues in the conclusion and future challenges section.

5.1 Preliminaries

Evolving access structure [100]: An evolving access structure $\mathcal{A} \in 2^{\mathbb{N}}$ is a monotone collection of subsets of the natural numbers such that for any $t \in \mathbb{N}$, the collection of subsets $\mathcal{A}_t = \mathcal{A} \cap [t]$ is an access structure.

Evolving Secret sharing scheme [100]: Let $\mathcal{A} = \{\mathcal{A}_t\}_{t\in\mathbb{N}}$ be an evolving access structure. Let S be a domain of secrets, where $|S| \geq 2$. A secret sharing scheme $\mathcal{S}$ for $\mathcal{A}$ and S consists of a pair of algorithms $(SHARE; RECON)$. $SHARE$ is probabilistic sharing algorithm and $RECON$ is a deterministic reconstruction algorithm which realize the secret sharing scheme.

Dynamic thresholds [100]: A dynamic threshold access structure has a sequence $k_1 \leq k_2 \leq \ldots$ of positive integers. For any $t \in \mathbb{N}$, the set of qualified sets at time t contains all those sets of cardinality at least k_t. Of particular interest is the sequence with $k_t = \gamma \cdot t$ where $\gamma \in (0, 1)$ is a fixed constant.

Quantum one-time pad - (QOTP)[50, 119]: The QOTP is a symmetric-key encryption scheme. Key is generated classically by picking $a_i, b_i \in_R \{0, 1\}$. A sequence of qubits is encrypted qubit-by-qubit by applying $X^{a_i}Z^{b_i}$ to the i-th qubit (X and Y are *Pauli Gates*). Qubit-by-qubit decryption is done by applying $X^{a_i}Z^{b_i}$ to the i-th qubit. Quantum one time pad provides information-theoretic security [31].

Trap Code[30, 32]: A trap code is a quantum message authentication code. It ensures integrity of the data by combining a CSS code to spread out the data with the insertion of check qubits (traps) at random locations, and a quantum one-time pad on the entire state. Key generation is done by choosing a random permutation σ. Qubit-by-qubit

encoding is done by applying E(a quantum error correcting(QECC) CSS code), appending traps: half $|0\rangle$ (the X-traps), half $|+\rangle$ (the Z-traps), permuting the qubits by σ and applying a quantum one-time pad using the classical randomness in the key. To decode, remove the quantum one-time pad, apply inverse permutation σ^{-1} and measure X-traps in the computational basis and the Z-traps in the Hadamard basis. If they are not in their original state, reject. Finally decode the QECC. The security of the trap code depends on the distance d of the underlying QECC. For more on QECCs, we refer the reader to [79, 80]. One may also use the Clifford code [2].

5.2 Quantum Evolving Secret Sharing Scheme

• We use dynamic thresholds(thresholds increase as new participants arrive) for our scheme. A threshold is the minimum number of participants needed to be present in a qualified set.

• The participants are first grouped into generations of sizes in an increasing order. Generations are denoted by G_i, $i = 1, 2, 3, \ldots$ and their size is denoted by $|G_i|$. We set $|G_1| < |G_2| < |G_3| < \ldots$.

• Participants arrive one by one and each arriving participant Bob_j is assigned a generation. When a generation is exhausted, a new generation is created. A discussion on the sizes of the generations is given in section 3.3.

• **An example** : Suppose at time t we have 3 generations G_1, G_2, G_3 with

$$|G_1 = 3|, |G_2| = 5, |G_3| = 8.$$

Let

1. $G_1 = \{1_{(1,1)}, 2_{(2,2)}, 3_{(3,3)}\}$.
2. $G_2 = \{4_{(4,1)}, 5_{(5,2)}, 6_{(6,3)}, 7_{(7,4)}, 8_{(8,5)}\}$.
3. $G_3 = \{9_{(9,1)}, 10_{(10,2)}, 11_{(11,3)}, 12_{(12,4)}, 13_{(13,5)}, 14_{(14,6)}, 15_{(15,7)}\}$.

Suppose a new participant arrives. Note that the generation G_3 has not been exhausted yet. So the new participant will be added to G_3. The new participant is denoted as $16_{(16,8)}$. The first subscript denotes the index of the participant since the beginning and the second subscript denotes the index of the participant in the group G_3. Now G_3 is exhausted. So when a new participant arrives, it will be assigned to a new generation G_4 and it will be denoted by $17_{(17,1)}$ since it is the overall 17-th participant and the first participant in G_4.

• *Sharing Secret*: To share a secret, Alice shares the secret state to the generation G_1, a suitably modified secret to the second generation G_2 and so on.

• To reconstruct the secret, participants start recovering the secrets of the generations starting from the presently last generation. Then using the recovered secret, the secret of the presently second-last generation is recovered and so on. This iterative process goes on and stops with $G_3, ..., G_2, ..., G_1$. When the first generation is reached, the initial secret is recovered.

Reason to consider dynamic thresholds

Let us suppose that we have a (k, n) quantum threshold secret sharing scheme (QTSSS) where a secret quantum state is shared among n participants and any k or more of those

participants can reconstruct the secret. In [51], the authors show that due to the "*no-cloning theorem*" it must hold that $k > n/2$, otherwise two disjoint sets of participants can reconstruct the secret separately and can create two separate copies of the secret quantum state which violates the "*no-cloning theorem*". Hence this puts a barrier for constructing the quantum version of an evolving threshold secret scheme [99]. In [99], the authors construct a classical k-threshold evolving secret sharing scheme. Here the participants arrive one by one and at any time any k-participants can reconstruct the secret. However, while trying to share a quantum secret via this scheme, the "*no-cloning theorem*" poses a barrier. At some time the total number of participants present can become more than $2k$ and two disjoint copies of participants can reconstruct two copies of the unknown quantum secret violating the "*no-cloning theorem*". Hence we consider dynamic thresholds as in [100], where not only the number of participants increases with time but the thresholds increase with time. Taking a proper increasing sequence of dynamic thresholds, we need to ensure that at any time t when the number of participants is n_t and the threshold is k_t, we have

$$k_t > n_t/2.$$

Hence in the following construction we shall construct a *quantum evolving secret sharing scheme with dynamic thresholds* to share a quantum secret.

5.2.1 Main Construction

The dealer Alice generates a sequence of unknown quantum states as secret $|S\rangle = |\phi_1\rangle, ..., |\phi_m\rangle$, where $|\phi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$, $(i = 1, \ldots, m)$, to be shared among the parties.

When a generation G_g begins, Alice recalls state $|S_A\rangle$ for each tuple $A = (c_0, \ldots, c_g)$ where $0 \leq c_i \leq |G_i|$, $(i = 1, \ldots, g)$. c_i denotes the number of parties from generation G_i who would take part in secret reconstruction procedure. A party in generation G_g has two identities attached to it – (1) i, its index in generation G_g and (2) $Id_{(i,g)}$, the index of the party since the start of the sharing process.

Notation: Denote the sharing and reconstruction procedure of a (k, n)-QTSS as $Sh_{Th}(k, n)$ (the sharing procedure) and $Rec_{Th}(k, n)$(the reconstruction procedure).

Notation: Denote the sharing and reconstruction procedure of the main quantum evolving scheme as $SHARE$ (the sharing procedure) and $RECON$(the reconstruction procedure).

• **Share distribution protocol**: Suppose at the current time t, let there be g generations for which the shares are already distributed. To share the secret $|S\rangle$in the generation G_{g+1}, Alice runs **Algorithm 1**. Define:

- For $A = (c_1, \ldots, c_g)$, define $prev(A) := (c_1, \ldots, c_{g-1})$.
- $\left|S_{prev(A)}\right\rangle := |S\rangle$ if $g = 1$.
- $\left|S_{prev(A)}\right\rangle := \left|S_{(c_1,\ldots,c_{g-1})}\right\rangle$ if $g > 1$.

• **Algorithm 1** : $[SHARE\ |S\rangle]$

1. *For every tuple* $A = (c_1, \ldots, c_{g+1})$ *do steps 2 TO 5.*
2. *For each* c_j, $1 \leq j \leq g + 1$ *do steps 3 TO 5.*
3. *Set* $A_j \leftarrow (c_1, \ldots, c_j)$.

4. *If* $c_j = 0$, *set* $\left|S_{A_j}\right\rangle \leftarrow \left|S_{prev(A_j)}\right\rangle = \left|S_{(c_1,...,c_{j-1})}\right\rangle$ *and Stop.*

5. *If* $c_j > 0$ *do the following :*

   (a) *Do* ***Procedure****: ShareinGen*$(\left|S_{prev(A_j)}\right\rangle, j)$ *to share* $\left|S_{prev(A_j)}\right\rangle$ *in generation* G_j.

   (b) *Get a random state* $\left|r_{A_j}\right\rangle = \left|r_{(c_1,...,c_j)}\right\rangle$ *from* ***Procedure****: RandomShareGen*(A_j, c_j).

   (c) *Do* ***Procedure****: UpdateSecret*$(\left|S_{prev(A_j)}\right\rangle, \left|r_{A_j}\right\rangle)$ *to get the state* $\left|S_{A_j}\right\rangle$ *which will be used for the next generation*

6. STOP.

1. **Procedure**: *ShareinGen*$(\left|S_{prev(A_j)}\right\rangle, j)$

2. *For each party* $i \in G_j$*: do steps 3 to 4.*

3. *Share* $\left|S_{prev(A_j)}\right\rangle$ *via a* $Sh_{Th}(k_{Id_{(i,j)}} - \Sigma_{k=1}^{j-1} c_k, i)$ *quantum threshold scheme to get shares* $|Bob_1\rangle, ..., |Bob_i\rangle$.

4. *For each* p, $1 \leq p \leq i$, *give share* $|Bob_p\rangle$ *to the* p*-th participant in the generation.*

5. STOP.

1. **Procedure**: *RandomShareGen*(A_j, c_j)

2. *Prepare* $2(g+1)$ *traps: half* $|0\rangle$ *(the X-traps), half* $|+\rangle$ *(the Z-traps). Call the state* $|Temp\rangle$.

3. *Permute* $|Temp\rangle$ *by a random permutation* σ_1 *and call the resulting state*

$\left|r_{A_j}\right\rangle = \left|r_{(c_1,...,c_j)}\right\rangle$.

4. *Share* $\left|r_{A_j}\right\rangle$ *via an* $Sh_{Th}(c_j, |G_j|)$ *quantum threshold scheme among the participants of generation* G_j.

5. *Return* $\left|r_{A_j}\right\rangle$ *to Algorithm 1.*

6. STOP.

1. **Procedure**: *UpdateSecret*$(\left|S_{prev(A_j)}\right\rangle, \left|r_{A_j}\right\rangle)$

2. *On* $\left|S_{prev(A_j)}\right\rangle$ *apply* E*, an encoding operation for a quantum error correcting CSS code to get the state* $\left|S_{temp(A)}\right\rangle$.

3. *To* $\left|S_{temp(A)}\right\rangle$*, append* $\left|r_{A_j}\right\rangle$*. Permute all the qubits according to a random permutation* σ_2*. Finally apply a quantum one-time pad using the classical randomness in the key. Assign the resulting as* $\left|S_{A_j}\right\rangle = \left|S_{(c_1,...,c_j)}\right\rangle$.

4. *Return* $\left|S_{A_j}\right\rangle$ *to algorithm 1.*

5. STOP.

• **Secret reconstruction protocol**: The reconstruction procedure is as follows: Let us suppose that at time t, we have a qualified set A_t. For this set let participants come from atmost g generations.

• **Algorithm 2** : [RECON : Reconstruction from g generations]

1. *Recall corresponding tuple* $A = (c_1, ..., c_g)$ *for* A_t *with* c_i *participants coming from generation* G_i*,* $1 \leq i \leq g$ *and* G_g *is the last generation from where participants arrive.*

2. *Recover the states* $\left|r_{(c_1)}\right\rangle$, $\left|r_{(c_1,c_2)}\right\rangle$,...,$\left|r_{(c_1,...c_g)}\right\rangle$ *by the reconstruction procedure of the* $Rec_{Th}(c_i, |G_i|)$ *quantum threshold scheme.(*$1 \leq i \leq g$*)*

3. *Assign* $f \leftarrow$ *number of participants out of the* c_g *participants coming from the first* i *participants.*

4. *In generation* G_g *recover* $\left|S_{(c_1,...,c_{g-1})}\right\rangle = \left|S_{prev(A)}\right\rangle$ *via the reconstruction procedure* $Rec_{Th}(k_{Id_{(i,g)}} - \Sigma_{i=1}^{g-1} c_i, i)$ *quantum threshold scheme.*

5. *From* $\left|S_{(c_1,...,c_{g-1})}\right\rangle = \left|S_{prev(A)}\right\rangle$ *and* $\left|r_{(c_1,...c_g)}\right\rangle$, *recover the secret* $\left|S_{(c_1,...,c_{g-2})}\right\rangle = \left|S_{prev(prev(A))}\right\rangle$ *using* ***Procedure****:DecodeSecret(*$\left|S_{prev(A)}\right\rangle, \left|r_{(c_1,...c_g)}\right\rangle$*).*

6. *Continuing in this way, recover the states* $\left|S_{(c_1,...,c_{g-3})}\right\rangle$, $\left|S_{(c_1,...,c_{g-4})}\right\rangle$,..*all the way to* $\left|S_{(1)}\right\rangle$ *which is the secret* $|S\rangle$*.*

7. STOP.

1. **Procedure**: *DecodeSecret(*$\left|S_{prev(A)}\right\rangle, |r_A\rangle$*).*

2. *On* $\left|S_{(c_1,...,c_{g-1})}\right\rangle$ *remove the quantum one-time pad and apply the inverse permutation* σ_2^{-1}*.*

3. *On the resulting state, replace the last* $2m$ *states by* $\sigma_1^{-1}(\left|r_{(c_1,...c_g)}\right\rangle)$ *to get* $\left|S_{temp(prev(A))}\right\rangle$*.*

4. *Decode* $\left|S_{temp(prev(A))}\right\rangle$ *using the quantum error correcting code to obtain* $\left|S_{(c_1,...,c_{g-2})}\right\rangle$*.*

5. **End Procedure**: *DecodeSecret*

5.2.2 Secret recovery and measurement schemes

The constructed scheme uses quantum threshold schemes as building blocks. In this chapter we do not consider the problem of quantum threshold secret sharing, and simply assume that it can be done when necessary [29,31]. Our scheme inherits the properties of the underlying threshold schemes being used and hence the measurement schemes of the overall quantum evolving secret sharing scheme is the same as the one used as the underlying threshold scheme. We note that in our construction, the secret quantum state is recovered by a recursive procedure, where the share reconstruction procedure for the quantum threshold scheme is applied several times. Now since we are working with quantum secrets instead of classical secrets, we work with underlying quantum threshold schemes where the quantum secret is recovered by phase shift operations combined with Lagrange's interpolation [131] and avoid measurement based quantum threshold schemes. This is due to the fact that after a measurement operation is done we will find the system to be in one of these basis states, even though it may have been in any state before the measurement. The only measurement operations we do are to decode the trap code and to check the integrity of the share by inserting decoy states while sending quantum data from Alice to the participants. Here the measurement does not affect the quantum secret at any stage and hence the recursive procedure can continue. In this section we briefly describe one such scheme [131] which can be used as the underlying quantum threshold secret sharing scheme.

Brief description of quantum threshold scheme [131] used as $Sh_{Th}(k,n)$ and $Rec_{Th}(k,n)$

• Distribution of private keys

1. Alice selects a polynomial f of degree $k-1$ in an appropriate finite field and evaluates the polynomial at n different x_i's.

2. Alice publishes the x_i's and shares the $f(x_i)$'s to the n, participants via quantum secure direct communication and this $f(x_i)$'s are the private keys of the participants.

• Sharing quantum state

1. Alice generates the quantum states as secret $|S\rangle$ (see section 3.1).

2. Alice applies phase shift $U(\theta)$ on every quantum state in the sequence, where $U(\theta) = \cos(\theta)\,|0\rangle\langle 0| - \sin(\theta)\,|0\rangle\langle 1| + \sin(\theta)\,|1\rangle\langle 0| \cos(\theta)\,|1\rangle\langle 1|$, where $\theta = 2\pi - \frac{S}{N}$ with appropriately chosen S and N owing to security requirements of the system.

3. After applying the phase shift, Alice sends the sequence to one participant say Bob_1.

• Secret recovery

1. Bob_1 performs phase-shift $U(\theta_1)$ on every state in the sequence with $\theta_1 = \frac{L_1 f(x_1)}{N}$, where $L_1 = \Pi_{1\leq j\leq k, j\neq 1} \frac{x_j}{x_j - x_1}$ and sends the resulting sequence to Bob_2.

2. Bob_2 performs phase-shift $U(\theta_2)$ on every state in the sequence with $\theta_2 = \frac{L_2 f(x_2)}{N}$, where $L_2 = \Pi_{1\leq j\leq k, j\neq 2} \frac{x_j}{x_j - x_2}$ and sends the resulting sequence to Bob_3 and so on.

3. After Bob_k applies the phase shift $U(\theta_k)$, the resulting sequence is the required recovered secret. For a detailed proof, we refer the reader to [131].

Measurement schemes used in the constructions

Measurements are used in two different cases in our construction.

1. Whenever secret is sent from the dealer Alice to the participants Bob's or from one participant to the other, decoy particles $|0\rangle, |1\rangle$, $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, $|-\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ are inserted in the sequence for eavesdropping detection. The initial state and the positions of these decoy particles are remembered by the sender. The receiver measures these particles in the appropriate bases as announced by the sender and the receiver announces the measurement results. The sender computes the error rate by comparing the measurement results with the initial states. If the error rates exceeds certain threshold, the process is aborted.

2. Measurements can be done for security of the system during the decoding process of the trap code used in step 5 of Algorithm 2 and in the procedure *DecodeSecret*. To decode a trap code, the X-traps are measured in the computational basis and the Z-traps are measured in the Hadamard basis. If they are not in their original state, they are rejected.

5.2.3 Correctness and Privacy

• **Correctness of Algorithm 1 and 2 and the scheme.** For g generations, Alice has to maintain tuples $(c_1, ..., c_g)$ for all combinations of the numbers of participants arriving from each generation. The procedure ShareinGen takes the previous secret, and

shares it among certain participants of the generation via a quantum threshold scheme as per the dynammic threshold. The procedure RandomShareGen(j, c_j) prepares the random state $\left|r_{(c_1,\ldots,c_j)}\right\rangle$. This is done by mixing $2m$ $|0\rangle$ states and $|1\rangle$ states via a random permutation σ_1. This random permutation has to be shared classically for the reconstruction procedure. Finally $\left|r_{(c_1,\ldots,c_j)}\right\rangle$ is shared among all the parties of the current generation via a quantum threshold scheme. Finally the procedure UpdateSecret updates the secret $\left|S_{prev(A)}\right\rangle$ using $\left|r_{(c_1,\ldots c_g)}\right\rangle$. First $\left|S_{prev(A)}\right\rangle$ is encoded through a QECC (CSS) and $\left|r_{(c_1,\ldots c_g)}\right\rangle$ is appended to it. The resulting state is mixed via another random permutation σ_2 (shared classically) and a quantum one-time pad. Without applying σ_1, the state $\left|r_{(c_1,\ldots c_g)}\right\rangle$ is not a random state which is required for security.

A qualified set of participants recovers the random states $\left|r_{(c_1)}\right\rangle$, $\left|r_{(c_1,c_1)}\right\rangle$,..., $\left|r_{(c_1,\ldots,c_g)}\right\rangle$ from each generation. Since the set is qualified, they satisfy the dynamic threshold in the last generation G_g from which participants arrive. Hence they can recover the state $\left|S_{prev(A_g)}\right\rangle$. Reconstruction follows from the reconstruction procedure of the quantum threshold schemes used. Using $\left|S_{prev(A_g)}\right\rangle$ and $\left|r_{(c_1,\ldots,c_g)}\right\rangle$, the participants recover $\left|S_{prev(A_{g-1})}\right\rangle$ and so on all the way to $\left|S_{(c_1)}\right\rangle$ which is the secret $|S\rangle$.

• **Privacy of the scheme.** A forbidden set has the following property – there exists at least one generation form which sufficiently many participants as per the dynamic thresholds do not arrive. The arriving participants of that particular generation cannot reconstruct the corresponding $\left|S_{prev(A)}\right\rangle$ by the security of the threshold scheme. Hence they cannot remove the masking of the secret to reveal the secret of the previous generation and finally cannot reconstruct the secret $|S\rangle$. Hence the construction is private.

From the above discussion we have the following theorem:

Theorem 1: The $SHARE$ and $RECON$ procedures construct a quantum evolving secret sharing scheme which shares a quantum secret $|S\rangle$ among unbounded number of participants with an increasing sequence of thresholds.

5.2.4 Sizes of the generations, dimensions of shares

The dimension of the share of a participant is estimated in three parts. From section 3.2.1, we shall assume that the dimension of the share resulting from the quantum threshold schemes is same as the dimension of the secret. Also note that applying a quantum one-time pad does not change the dimension. Let us suppose that a participant is in generation G_g.

1. *From procedure ShareinGen* : The contribution in the dimension of the share from this procedure is bounded by

$$(\prod_{j=1}^{g} |G_j|) \times \dim(share_{G_g}),$$

where $\dim(share_{G_j})$ denotes the dimension of a share in generation G_g.

2. *From procedure RandomShareGen* : The second contribution in the dimension of the share from this procedure is again bounded by

$$(\prod_{j=1}^{g} |G_j|) \times \dim(random_{G_g}),$$

where $\dim(random_{G_g})$ denotes the denotes the dimension of a random state in generation G_g.

3. *Estimates of dim($share_{G_g}$) and dim($random_{G_g}$)* : From the procedure *RandomShareGen*, the dimension of the random share is $2(g+1)$ and from the procedure *UpdateSecret* the dimension of the share is bounded by $\mathcal{E}^g \times 2g$, where $\mathcal{E}$ denotes the factor by which data is spread by the encoding CSS code.

By multiplying items 1,2 and 3, we find that the dimension of the share of a participant in generation G_g is bounded by $O((\prod_{j=1}^{g} |G_j|)^2 \times \mathcal{E}^g \times g^2)$. Now if we take the size of generation G_j to be 2^{2^j}, then we have the dimension of the share of a participant in generation G_g to be bounded by

$$O((\prod_{j=1}^{g} 2^{2^j})^2 \times \mathcal{E}^g \times g^2)$$

$$= O((2^{\sum_{j=1}^{g} 2^j})^2 \times \mathcal{E}^g \times g^2)$$

$$= O((2^{2^{2g+1}})^2 \times \mathcal{E}^g \times g^2).$$

Now if the overall index of a participant is t, then it belongs to the generation $G_{\lceil \log \log t \rceil}$. Hence the dimension of the t-th participant is bounded by

$$O(t^4 \log t (\log \log t)^2).$$

A reduction in the number of traps reduces the dimension to some extent.

5.2.5 Memory usage

The dealer Alice needs to maintain information for all the tuples. For g generations there are $\prod_{j=1}^{g}(|G_j|+1)$ tuples. For each of these tuples Alice needs to remember

$|G_j|$ random permutations σ_1 per generation G_j. Again quantum memory is needed to remember the random permutations σ_2's. Overall an exponential amount of quantum memory is needed for this process. Moreover if decoy states are to be used for security, the participants need to remember the positions where the decoy particles are inserted which adds to the required memory.

Possible Improvements

• This construction can be generalized further to weighted schemes. Let us suppose that the each participant in a generation G_g carries some weight w_g. The condition for being a qualified set is as follows: the weighted sum of the participants in a set must be more than the current threshold. For this Alice instead of maintaining tuples $(c_1, ..., c_g)$, maintains tuples $(w_1c_1, ..., w_gc_g)$ and with these tuples it proceeds with the algorithms. The remaining construction is unchanged.

• The major drawback of this construction is the huge amount of quantum memory needed by Alice for the tuples $(c_1, ...c_g)$. But note the following: Suppose $(c_1, ..., c_g)$ is a tuple corresponding to a qualified set at time t. This means that $\Sigma_{i=1}^{g} c_g \geq k_t$. Consider another tuple $(p_1, ..., p_g)$ at the same time such that $p_i \geq c_i \ \forall \ i$. This means that the tuple $(p_1, ..., p_g)$ is also qualified at time t. Hence it is enough to only maintain the tuple $(min(c_1), ..., min(c_g))$ such that $\Sigma_{i=1}^{g} c_g \geq k_t$. This reduces the usage of quantum memory to some extent. Also when the threshold is updated after some time some qualified set may become unqualified. Hence the quantum memory needs to be updated/ refreshed after some time. More on reducing quantum memory is discussed in section 6.

• The crucial idea of the construction is to divide the participants into several generations. Whenever too many participants arrive a new generation is created. Here consider one way to reduce the number of tuples based on a simple combinatorial observation. Let us suppose that we have g generations, $G_1, ..., G_g$. The tuples are of the form $(c_1, ..., c_g)$. So the number of such tuples the dealer has to maintain is $(|G_1| + 1) \times ... \times (|G_g| + 1)$. Suppose that the generations $G_1, ..., G_g$ are squashed into one single generation $\mathcal{G}$ containing $(G_1| + ... + |G_g|)$ participants, then the dealer has to maintain the tuple $(\mathcal{C}_1)$. Clearly now the dealer has to maintain far less tuples than in the previous case and the process may continue by adding participants to a new generation $\mathcal{G}_2$. However the threshold conditions change. If we allow this change then this method can reduce the usage of quantum memory to a considerable extent.

• Our method in section 4 also helps to reduce memory usage to a certain extent.

5.2.6 Error-tolerance

The security of the trap code depends on the distance d of the underlying QECC. It is $(2/3)^{d/2}$-secure against Pauli attacks, that is the probability (taken over all possible permutations / QOTPs) that a fixed Pauli attack Q acts non-trivially on the logical data without revealing an error in the traps is at most $(2/3)^{d/2}$. The underlying code we have used is a $[[m, 1, d]]$- quantum error-correcting CSS code. Hence for each participant it can tolerate up to d-errors.

5.2.7 Discussions

In [99] and [100], the authors construct classical threshold evolving secret sharing schemes and evolving secret sharing schemes with dynamics thresholds respectively. But through these schemes secret cannot be protected in a quantum environment. Our construction overcomes this. As the authors note in [131] it is harder to construct a secret sharing scheme which shares a quantum secret and our scheme can handle quantum secrets. While the "no-cloning" theorem makes it impossible to construct a fixed-threshold quantum evolving scheme sharing a quantum secret, there is no problem to share a classical secret through a quantum evolving secret sharing scheme, for fixed threshold or dynamic threshold. The fixed threshold quantum evolving scheme can be taken as a special case of the dynamic case with all thresholds $k_t = k$. Our scheme, combined with superdense coding reduces the share size by half as compared to classical evolving schemes for sharing classical secrets.

5.3 Comparison

In this section we draw comparisons of our scheme with some of the nearby constructions of schemes which can accommodate new participants over. The main advantage of this scheme compared to the existing quantum dynamic secret sharing schemes is that without secret redistribution, we are able to accommodate unbounded number of new participants. To the best of our knowledge no such construction so far can handle this scenario. Decoy states have been used previously to achieve security against popular attacks such as intercept-and-resend attack, entangle-and-measure attack, man-

in-the-middle attack, trojan horse attacks [128, 131] and resilience as in [113]. Since we repeatedly use the cited quantum threshold schemes, our scheme inherits security against such attacks.

Chapter 6

A Quantum Hierarchical Secret Sharing Scheme - Further Studies

In this chapter we construct a quantum hierarchical secret sharing scheme (QHSSS) which can handle an arbitrary number of groups and different conditions for hierarchy. Our construction follows the ideas of chapter 5 with suitable modifications, is flexible and with slight modifications to the scheme, we are able to realise quantum versions of weighted threshold access structures, uniform multipartite access structures and partially hierarchical access structures. Quantum computation has grown tremendously in the last two decades and we build on this progress by utilizing quantum threshold schemes[14, 131, 128] and quantum error correcting codes repeatedly in our construction.

We cover a lot of access structures and to our best knowledge quantum schemes for some of the access structures have not been constructed before. Hence this work conducts an extensive study of quantum schemes for multipartite access structures.

6.1 Preliminaries

An access structure is *multipartite* if the participant set $\mathcal{P}$ can be partitioned into several disjoint groups. Each group of participants has distinct properties and the

shared secret is reconstructed according to those properties. We shall look at some multipartite access structures.

• Let the set of participants $\mathcal{P}$ be partitioned into several groups P_i such that $\mathcal{P} = \bigcup_{i=1}^{m} P_i$, where P_i's are non-empty, and $P_i \bigcap P_j = \phi$ for any distinct i, j.

The following access structures are of interest:

• **Weighted threshold access structure**[118] : Each participant is assigned a weight in such a way that the participants with the same weight belong to the same group and a subset is qualified if and only if the weighted sum of its members is greater than some given threshold.

• **Compartmented access structures**[142] : Each group (compartment) P_i $(1 \leq i \leq m)$ has a threshold t_i $(1 \leq i \leq m)$ and the qualified subsets require all the m thresholds are satisfied.

• **Uniform threshold access structures**[142] : Each group P_i $(1 \leq i \leq m)$ has a threshold t_i $(1 \leq i \leq m)$ and the qualified subsets require at least k $(1 \leq i \leq m)$ thresholds t_i's should be satisfied.

• **Uniform multipartite access structures**[90] : Every qualified subset comprises at least t participants from no less than k $(1 \leq i \leq m)$ groups.

• **Hierarchical threshold access structures**[142, 147] : All the participant groups P_i's $(1 \leq i \leq m)$ are pairwise hierarchically comparable and the thresholds satisfy $t_1 \leq t_2 \leq ... \leq t_m$. In fact, all the groups form a totally ordered set according to their hierarchies. A participant at a lower level can be replaced by a higher level participant.

• **Quasi-threshold multipartite access structures**[122] : Each group P_i has a threshold t_i $(1 \leq i \leq m)$, and a special participant p is contained in some group, w.l.o.g., P_1. There are two cases such that the secret can be reconstructed, that is, a set of at least t_1 participants from $P_1 \backslash \{p\}$ is qualified; otherwise, a set contains at least $(t_1 - 1)$ participants from $P_1 \backslash \{p\}$ and no less than k $(1 \leq k \leq m-1)$ other groups that achieve their thresholds are also qualified.

• **Partially hierarchical access structures**[70] : Hierarchical relationships among groups can be seen as a star-like partial order, that is, there exists a higher level group on the center node of a star, and the other groups are placed on the leaf nodes.

• In our constructions, we shall repeatedly use a (k, n) quantum threshold secret sharing scheme(QTSS), which realizes a (k, n)-threshold access structure, where a quantum secret is shared among n participants and any k or more participants can reconstruct the secret and sets of participants with less than k participants have no information about the secret.

• **Notation :** (k, n)-QTSSS denotes a k-out-of-n quantum threshold secret sharing scheme. We shall not go into the exact details and constructions of these schemes and refer the reader to [128, 131] for the details and security of these schemes.

To generate shares for the participants we shall use quantum one-time pad and trap codes as in chapter 5.

6.2 Technical Details

6.2.1 Warm-up: Comparmented, Uniform threshold access sets

The dealer/boss Alice generates a sequence of unknown quantum states as secret $|S\rangle = |\phi_1\rangle, ..., |\phi_h\rangle$, where $|\phi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$, $(i = 1, \ldots, h)$, to be shared among the participants. Suppose that the participants are grouped into disjoint sets P_1, P_2,...,P_m.

Compartmented access structures

Recall that each group (compartment) P_i $(1 \leq i \leq m)$ has a threshold t_i $(1 \leq i \leq m)$ and the qualified subsets require all the m thresholds are satisfied.

• **Notation :** (k, n)-QTSS denotes a k-out-of-n quantum threshold secret sharing scheme.

• **Notation :** $|P_i|$ denotes the number of participants in the group P_i.

Secret Sharing goes on in two stages :-

• **Preparing share states**:

1. Alice first prepares shares $|S_1\rangle$,...,$|S_m\rangle$ from $|S\rangle$-via a (m, m)-QTSS.
2. For each $|S_i\rangle$, $(1 \leq i \leq m)$, Alice then prepares $|P_i|$ states $|S_i^1\rangle, ..., \left|S_i^{|P_i|}\right\rangle$ via a $(t_i, |P_i|)$-QTSS.
3. Alice assigns the share $\left|S_i^j\right\rangle$ to the j-th participant in the group P_i, $(1 \leq j \leq |P_i|)$ and $(1 \leq 1 \leq m)$.

• **Secret Reconstruction:**

1. For each i, $(1 \leq 1 \leq m)$, t_i participants in the group P_i reconstruct $|S_i\rangle$ from the shares $|S_i^1\rangle, ..., \left|S_i^{|P_i|}\right\rangle$ via the reconstruction procedure of the $(t_i, |P_i|)$-QTSS.

2. After recovering all the $|S_i\rangle$'s, the secret $|S\rangle$ can be reconstructed via the reconstruction procedure of the (m, m)-QTSS.

Since an (m, m)-QTSS is used, all the thresholds t_i's must be satisfied and all the $|S_i\rangle$'s are required to reconstruct the secret $|S\rangle$. The correctness, privacy and the security of the above scheme against popular quantum attacks follows from that of the $(t_i, |P_i|)$-QTSS and (m, m)-QTSS. More details are give in section 3.2.

Uniform threshold access structures

Recall that for this access structure at least k of the thresholds t_i's should be satisfied. Hence instead of using an (m, m)-QTSS, Alice first prepares shares $|S_1\rangle$,...,$|S_m\rangle$ from $|S\rangle$-via a (k, m)-QTSS. Then she shares each $|S_i\rangle$ in the group P_i via a $(t_i, |P_i|)$-QTSS as before. During the reconstruction procedure at least k of the $|S_i\rangle$'s are required to reconstruct the secret $|S\rangle$. Hence atleast k of the thresholds t_i's need to be satisfied. Again the correctness and the security of the above scheme follows from that of the $(t_i, |P_i|)$-QTSS and (k, m)-QTSS. More details in section 6.2.2.

6.2.2 Main construction : Hierarchical Threshold Access Structures

In a hierarchical threshold access structure all the participant groups P_i's $(1 \leq i \leq m)$ are pairwise hierarchically comparable and the thresholds satisfy $t_1 \leq t_2 \leq ... \leq t_m$. In fact, all the groups form a totally ordered set according to their hierarchies. Let us suppose that the groups P_i's are arranged in the following total hierarchical order $P_1 \geq P_2 \geq ... \geq P_m$. This means that the participants of the group P_1 have higher power over the participants of the group P_2 and so on. In this construction we shall mirror the following real world scenario : the lowest level of agents reconstruct some state and send it to the next higher level of agents who in turn reconstructs a state and sends it to next higher level of agents in the hierarchy and so on. Finally the highest level agents collect the required states from the lower level agents and they reconstruct the secret.

• **Illustrative Example :** Suppose we have three groups P_1, P_2 and P_3. Let

$$P_1 = \{1, 2, 3, 4, 5\} \text{ with threshold } t_1 = 3$$

and

$$P_2 = \{6, 7, 8, 9, 10, 11, 12\} \text{ with threshold } t_2 = 5$$

and

$$P_3 = \{12, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23\} \text{ with threshold } t_2 = 7.$$

Condition for hierarchy : Atmost 5 participants can come from P_1, atmost 7 par-

ticipants can come from P_2 and 11 from P_3. So Alice maintains triples (p_1, p_2, p_3) for each combinations of the number of participants arriving from each group.

• To share a secret $|S\rangle$, Alice does the following :

For group P_1.

1. Share $|S\rangle$ among the participants of P_1.
2. Generate a random state $|r_1\rangle$.
3. Share $|r_1\rangle$ in the group P_1.

For group P_2.

1. Generate $|S_1\rangle = |S\rangle \maltese |r_1\rangle$, where $\maltese$ represent the operations (using trap codes and random permutations) to update $|S\rangle$ utilizing $|r_1\rangle$.
2. Share $|S_1\rangle$ among the participants of P_2.
3. Generate a random state $|r_2\rangle$ and share it among the participants of P_2.

For group P_3.

1. Generate $|S_2\rangle = |S_1\rangle \maltese |r_2\rangle = (|S\rangle \maltese |r_1\rangle) \maltese |r_2\rangle$.
2. Share $|S_2\rangle$ among the participants of P_3

• To reconstruct the secret,

1. Participants in group P_3 recover $|S_2\rangle$.

2. Participants in group P_2 recover $|r_2\rangle$.

3. Using $|S_2\rangle$ $(= |S_1\rangle \maltese |r_2\rangle)$ and $|r_2\rangle$, $|S_1\rangle$ is recovered.

4. Participants in group P_1 recover $|r_1\rangle$.

5. Finally using $|S_1\rangle$ $(= |S\rangle \maltese |r_1\rangle)$ and $|r_1\rangle$, the secret $|S\rangle$ is recovered.

The whole process is formalized below.

Share distribution protocol

From the groups P_i's various combinations of number of participants may arrive to present their shares. So Alice maintains tuples $(p_1, ..., p_m)$, where each p_i denotes the number of participants from each group P_i. Hence $0 \leq p_i \leq |P_i|$ $(1 \leq i \leq m)$. Again the dealer Alice generates a sequence of unknown quantum states as secret $|S\rangle = |\phi_1\rangle, ..., |\phi_h\rangle$, where $|\phi_i\rangle = \alpha_i |0\rangle + \beta_i |1\rangle$, $(i = 1, \ldots, h)$, to be shared among the participants. To share the secret $|S\rangle$, Alice runs **Algorithm 1**.

Define :

• For a tuple $T = (p_1, \ldots, p_k)$, define $T^{-1} := (p_1, \ldots, p_{k-1})$, $T^{-2} := (p_1, \ldots, p_{k-2})$ and so on.

• For a tuple $T = (p_1, \ldots, p_k)$, $|S_{T^{-1}}\rangle := |S\rangle$ if $k = 1$.

• For a tuple $T = (p_1, \ldots, p_k)$, $|S_{T^{-1}}\rangle := \left|S_{(p_1,\ldots,p_{k-1})}\right\rangle$ if $k > 1$.

Algorithm 1 : [Sharing secret $|S\rangle$]

1. *For every tuple* $T = (p_1, \ldots, p_m)$ *do steps 2 TO 5.*

2. *For each* j, $1 \leq j \leq m$ *do steps 3 TO 5.*

3. *Set* $T_j \leftarrow (p_1, \ldots, p_j)$.

4. *If* $p_j = 0$, *set* $\left|S_{T_j}\right\rangle \leftarrow \left|S_{T_j^{-1}}\right\rangle = \left|S_{(p_1,\ldots,p_{j-1})}\right\rangle$.

5. *If* $p_j > 0$ *do the following :*

   (a) *Share* $\left|S_{T_j^{-1}}\right\rangle$ *in the group* P_j. (Via algorithm 1a)

   (b) *Get a random state* $\left|r_{T_j}\right\rangle = \left|r_{(p_1,\ldots,p_j)}\right\rangle$ *and share it in the group* P_j (Via algorithm 1b)

   (c) *Update the secret state* $\left|S_{T_j^{-1}}\right\rangle$ *using* $\left|r_{T_j}\right\rangle$ *to get the state* $\left|S_{T_j}\right\rangle$. (Via algorithm 1c, this $\left|S_{T_j}\right\rangle$ will be used for the next group P_{j+1}.)

6. STOP

Recall that t_j is the threshold of the group P_j and p_j is the number of paticipants arriving from the group P_j.

Algorithm 1a : [Sharing $\left|S_{T_j^{-1}}\right\rangle$ in group P_j]

1. *For each participant* $i \in P_j$*: do steps 2 and 3.*

2. *Share* $\left|S_{T_j^{-1}}\right\rangle$ *via a* $(max(t_j, p_j), i)$*-QTSS to get shares* $|Bob_1\rangle, ..., |Bob_i\rangle$.

3. *For each* p, $1 \leq p \leq i$, *give share* $|Bob_p\rangle$ *to the* p*-th participant in the generation.*

4. STOP.

In the following algorithm 1b we generate a random state.

Algorithm 1b : [Get a random state $\left|r_{T_j}\right\rangle = \left|r_{(p_1,\ldots,p_j)}\right\rangle$]

1. *Prepare* $2m$ *traps: half* $|0\rangle$ *(the X-traps), half* $|+\rangle$ *(the Z-traps). Call the state* $|Temp\rangle$.
2. *Permute* $|Temp\rangle$ *by a random permutation* σ_1. *and call the resulting state* $\left|r_{T_j}\right\rangle = \left|r_{(p_1,\ldots,p_j)}\right\rangle$.
3. *Share* $\left|r_{T_j}\right\rangle$ *via a* $(t_j, |P_j|)$*-QTSS among the participants of generation* P_j.
4. *Return* $\left|r_{T_j}\right\rangle$.
5. STOP.

Finally in algorithm 1c, we update the secret for the next group.

Algorithm 1c :[Update the secret state $\left|S_{T_j^{-1}}\right\rangle$ using $\left|r_{T_j}\right\rangle$]

1. *On* $\left|S_{T_j^{-1}}\right\rangle$ *apply* E, *an encoding operation for a quantum error correcting* CSS *code to get the state* $|S_{current}\rangle$.
2. *To* $|S_{current}\rangle$, *append* $\left|r_{T_j}\right\rangle$.
3. *On the state obtained in step 2, permute all the qubits according to a random permutation* σ_2 *and finally apply a quantum one-time pad using the classical randomness in the key. Assign the resulting as* $\left|S_{T_j}\right\rangle = \left|S_{(p_1,\ldots,p_j)}\right\rangle$.
4. *Return* $\left|S_{T_j}\right\rangle$.
5. STOP.

Secret reconstruction protocol

Let us suppose that we have a qualified set Q. For this Q, we have the corresponding tuple $T_Q = (p_1, ..., p_m)$, where the p_i's are the number of participants arriving from the corresponding groups P_i's. Since Q is a qualified set the p_i's satisfy $p_i \geq t_i$ for each i $(1 \leq i \leq m)$.

Algorithm 2 : [Secret Reconstruction from set Q]

1. *Recall corresponding $T_Q = (p_1, ..., p_m)$ for Q with p_i participants coming from the group P_i, $1 \leq i \leq m$.*

2. *Recover the random states $\left|r_{(p_1)}\right\rangle$, $\left|r_{(p_1,p_2)}\right\rangle$,...,$\left|r_{(p_1,...p_m)}\right\rangle$ by the reconstruction procedure of $(p_i, |P_i|)$-QTSS($1 \leq i \leq m$).*

3. *In group P_m recover $\left|S_{(p_1,...,p_{m-1})}\right\rangle = \left|S_{T_m^{-1}}\right\rangle$ via the reconstruction procedure of the (t_m, p_m)-QTSS.*

4. *From $\left|S_{(p_1,...,p_{m-1})}\right\rangle = \left|S_{T_m^{-1}}\right\rangle$ and $\left|r_{(p_1,..,p_m)}\right\rangle$, recover the secret $\left|S_{(p_1,...,p_{m-2})}\right\rangle = \left|S_{T_m^{-2}}\right\rangle$.* (See algorithm 2a)

5. *Continuing in this way, recover the states $\left|S_{(p_1,...,p_{m-3})}\right\rangle$, $\left|S_{(p_1,...,c_{m-4})}\right\rangle$,..all the way to $\left|S_{(1)}\right\rangle$ which is the secret $|S\rangle$.*

6. STOP.

In the following algorithm 2a, from the states $\left|S_{T_m^{-1}}\right\rangle$ and $|r_{T_m}\rangle$, we recover the state $\left|S_{T_m^{-2}}\right\rangle = \left|S_{(p_1,...,p_{m-2})}\right\rangle$.

Algorithm 2a : [Secret recovery of previous group]

1. *On* $\left|S_{(p_1,...,p_{m-1})}\right\rangle$ *remove the quantum one-time pad and apply the inverse permutation* σ_2^{-1}.
2. *On the resulting state, replace the last* $2m$ *states by* $\sigma_1^{-1}(\left|r_{(p_1,...p_m)}\right\rangle)$ *to get* $|S_{current}\rangle$. (See steps 1 and 2 of algorithm 1c.)
3. *Decode* $|S_{current}\rangle$ *using the quantum error correcting code to obtain* $\left|S_{(p_1,...,p_{m-2})}\right\rangle$.
4. STOP.

Description and correctness of algorithms 1,1(a,b,c) and 2,2a

In this section we prove the correctness and privacy of our scheme through the descriptions of the algorithms.

• **Correctness :** We have groups of participants ordered as $P_1, ..., P_m$. Alice maintains tuples $(p_1, ..., p_m)$ for all combinations of the numbers of participants which may arrive from each group. Suppose Alice has a tuple $(p_1, ..., p_m)$. For this combination, Alice shares the secret $|S\rangle$ among the participants of the groups. For the group P_1, Alice shares the secret $|S\rangle$ among its participants(step 5a of algorithm 1, algorithm 1a). Furthermore it generates a random state $|r_{p_1}\rangle$ corresponding to p_1 and also shares it among the participants of the group P_1(step 5b of algorithm 1, algorithm 1b). For the next group P_2, Alice first updates the secret $|S\rangle$ using $|r_{p_1}\rangle$ to get a state $|S_{p_1}\rangle$(step 5c of algorithm 1, algorithm 1c) to be shared among the participants of P_2. Again Alice generates a random state and the process continues until all the groups have received the shares. Both the secret state and the random state is shared among the participants

via appropriate quantum threshold secret sharing schemes(QTSS). The random state is prepared via appending equal number of $|0\rangle$ states and $|+\rangle$ and permuting them using a random permutation σ_1(algorithm 1b). Secret $|S\rangle$ is updated(algorithm 1c) by first applying a quantum error correcting code(QECC) and appending it with the random state obtained from algorithm 1b. To the resulting state a random permutation is applied followed by a quantum one-time pad(QOTP) to get the updated secret. During the reconstruction procedure, each group recovers its random shares(algorithm 2). Since we have taken a qualified set, the number of participants from each group satisfy the thresholds and hence the group P_m can recover its secret $\left|S_{(p_1,\ldots,p_{m-1})}\right\rangle$. Using this $\left|S_{(p_1,\ldots,p_{m-1})}\right\rangle$ and the recovered random state $\left|r_{(p_1,\ldots,p_m)}\right\rangle$, algorithm 2a recovers the state $\left|S_{(p_1,\ldots,p_{m-2})}\right\rangle$ and so on all the way to $\left|S_{(p_1)}\right\rangle$ and the secret is reconstructed.

• **Privacy :** For privacy, we need a concrete characterization of forbidden sets for this access structure and we need to prove that a forbidden set has no information about the secret. In addition to this we shall prove that set which is not qualified cannot reconstruct the secret. In view of this we set the following :

Forbidden sets for the hierarchical access structure: A set of participants $A \subseteq \mathcal{P}$ is a forbidden set if the following condition is satisfied for all $1 \leq i \leq m$:

$$|A \cap P_i| < t_i,$$

i.e., none of the thresholds t_i's are satisfied. Note that if we denote the collection of all forbidden sets as Γ_{NO}, then it is easy to check that the collection of sets $2^{\mathcal{P}} \backslash \Gamma_{NO}$ is monotone.

Lemma 1: A forbidden set has no information about the secret.

Proof: Let A be a forbidden set. So for each group of participants P_i, $\leq i \leq m$, the number of participants present in A from P_i does not satisfy the threshold t_i,$1 \leq i \leq m$. Therefore for each i, by the underlying privacy of the (t_i, p_i)-QTSS, the participants have no information about each of $\left|S_{T_i^{-1}}\right\rangle$, i.e., for every two secrets quantum secrets, $|S_1\rangle$ and $|S_2\rangle$ and every possible vector of quantum shares $\langle|s_j\rangle\rangle_{p_j \in A \cap P_i}$,

$$Pr[\Pi_{A\cap P_i}(|S_1\rangle, |r\rangle) = \langle|s_j\rangle\rangle_{p_j\in A\cap P_i}] = Pr[\Pi_{A\cap P_i}(|S_2\rangle, |r\rangle) = \langle|s_j\rangle\rangle_{p_j\in A\cap P_i}].....(1)$$

where probability is over the choice of random states $|r\rangle$. Now the groups of participants P_i's are disjoint and A is an arbitrary forbidden set. Hence by taking product over all i in equation (1), we get that for every two secrets quantum secrets, $|S_1\rangle$ and $|S_2\rangle$ and every possible vector of quantum shares $\langle|s_j\rangle\rangle_{p_j\in A}$,

$$Pr[\Pi_{A}(|S_1\rangle, |r\rangle) = \langle|s_j\rangle\rangle_{p_j\in A}] = Pr[\Pi_{A}(|S_2\rangle, |r\rangle) = \langle|s_j\rangle\rangle_{p_j\in A}].....(2)$$

Thus from equation (2) we have privacy for our scheme. □

From the remark after definiton 2, for sets which are not qualified there are no requirements. However, we can prove the following:

Lemma 2: A set which is not qualified cannot reconstruct the secret.

Proof: Note that in our construction the group P_1 has the highest level participants. So if the participants in the highest level do not satisfy the threshold t_1, they cannot reconstruct the secret due the property of the QTSS. Let us suppose that the lowest level from which participants arrive is P_m. So if those participants satisty the threshold t_m, they can recover the random state and the secret of the previous group P_{m-1} and

the process continues till they recover the secret. But suppose for some group P_j $(1 \leq j < m)$, the threshold t_j is not satisfied, the process stops as the updates secret state of that group P_j cannot be recovered. In such a circumstance the reconstruction process cannot proceed and the secret cannot be recovered. □

Discussions

Note that the highest level participants can reconstruct the secret themselves. If the participants of the group P_2 want to recover the secret, at least t_2 participants need to produce their shares and they need to take the help of at least t_1 participants of the group P_1 to reconstruct the secret. Continuing in this way, for the participants of the group P_m, all the thresholds $t_m, ..., t_1$ should be satisfied to reconstruct the secret. Also note that if the threshold of the group P_m is not satisfied but all others $t_{m-1}, ..., t_1$ are satisfied, then the participants of the group P_{m-1} taking the help of the groups $P_{m-2}, ..., P_1$ can reconstruct the secret. This means that the group P_m can be replaced by participants of the group P_{m-1} and so on. While we have not imposed the condition for increasing thresholds an extra step can be added to algorithm 1 which ensures that the thresholds are in increasing order. Hence from the above discussion we have the following theorem 1.

Theorem 1: Algorithms 1,1a,1b,1c along with algorithms 2,2a construct a quantum hierarchical secret sharing scheme with secret $|S\rangle$ on groups $P_1, ..., P_m$ with thresholds $t_1, ..., t_m$.

Remarks

• Depending on the underlying QTSS scheme being used, the secret $|S\rangle$ can be a 2-level or a d-level secret.

• Note that it is not necessary for Alice to maintain all the tuples $(p_1, ..., p_m)$. Suppose two tuples $(p_1, ..., p_m)$ and $(q_1, ..., q_m)$ represent two qualified sets. Then the following tuple $(min(p_1, q_1), ..., min(p_1, q_1))$ also represents a qualified set. So for any qualified set it is enough to maintain the tuple $(t_1, ..., t_m)$, where the t_i's are the thresholds. Hence this drastically reduces the use of *quantum memory*. But it is required to maintain the tuples for unqualified sets because by our construction, starting from a group P_m, if the updated secret of the group $\left|S_{T_m^{-1}}\right\rangle$ is not recovered, then the final secret cannot be recovered even if the maskings (random states) of the previous groups are revealed.

6.3 Generalization to other schemes

In this section, we discuss how to construct quantum secret sharing schemes several multipartite access structures.

6.3.1 Weighted threshold access structures

In a weighted threshold access structure each participant is assigned a weight in such a way that the participants with the same weight belong to the same group and a subset is qualified if and only if the weight sum of its members is greater than some given threshold. In this access structure there is no hierarchical relation among the groups.

Let the disjoint groups be $P_1, ..., P_m$ with each participant in the group P_i having weight w_i $(1 \leq i \leq m)$. Let the threshold be t. In this case Alice maintains tuples $(p_1, ..., p_m)$ where $0 \leq p_i \leq |P_m|$. A set of participants is qualified if the corresponding tuple satisfies

$$\Sigma_{i=1}^{m} w_i p_i \geq t.$$

The secret sharing and the reconstruction algorithm is as follows :

Algorithm 3a : [Sharing secret $|S\rangle$]

1. *For every tuple* $T = (p_1, \ldots, p_m)$ *do steps 2 TO 7.*
2. *If* $\Sigma_{i=1}^{m} w_i p_i < t$ *share random states among the participants of the groups.*
3. *If* $\Sigma_{i=1}^{m} w_i p_i \geq t$ *do steps 4 to 7.*
4. *Count the number of* p_i*'s such that* $p_i > 0$*. Denote this number by* C*.*
5. *Prepare states* $|S_1\rangle, ..., |S_C\rangle$ *from* $|S\rangle$ *via a* (C, C)*-QTSS.*
6. *For each* p_i *such that* $p_i > 0$*, share one unassigned state* $|S_j\rangle$ *in the group* P_i *via a* $(p_i, |P_i|)$*-QTSS.*
7. *For each* p_i *such that* $p_i = 0$*, share random states among the participants of the group* P_i*.*
8. STOP

Algorithm 3b : [Secret Reconstruction from a qualified set Q]

1. *Recall the corresponding tuple* $(p_1, ..., p_m)$.

2. *Note the groups from which positive number of participants arrive i.e.,* $p_i > 0$.

3. *From the noted groups reconstruct the shares* $|S_1\rangle, ..., |S_C\rangle$ *via the reconstruction procedure of the corresponding* $(p_i, |P_i|)$*-QTSS.*

4. *Reconstruct the share from* $|S_1\rangle, ..., |S_C\rangle$ *via the reconstruction procedure of the* (C, C)*-QTSS.*

5. STOP.

• **Correctness**: Alice maintains tuples $(p_1, ..., p_m)$. If the weighted threshold condition is satisfied Alice counts (C) the number of groups from which positive number of participants arrive. Secret sharing goes on in two stages. First Alice prepares C states $|S_1\rangle, ..., |S_C\rangle$ by using a (C, C)-quantum threshold scheme. These states are shared among the groups from which positive number of participants arrive via $(p_i, |P_i|)$-quantum threshold schemes. To reconstruct the secret the corresponding groups reconstruct the shares $|S_1\rangle, ..., |S_C\rangle$ and finally the secret $|S\rangle$. Clearly by the construction, sets which are not qualified cannot reconstruct the secret.

• **Privacy**: A forbidden set in this access structure is the a set whose corresponding tuple $(p_1, ..., p_m)$ satisfies $\Sigma_{i=1}^{m} w_i p_i < t$. It is easy to check than the complement of the collection of all the forbidden sets is a monotone collection. By step (ii) of algorithm 3a, we have shared random states among the participants of the groups. Hence by the definition of privacy, a forbidden has no information about the secret state.

From the above discussion we have the following theorem.

Theorem 2: Algorithms 3a and 3b construct a quantum secret sharing scheme for weighted threshold access structures with secret $|S\rangle$ on groups $P_1, ..., P_m$ with weights $w_1, ..., w_m$ and threshold t.

• *Remarks*: Again it is possible to reduce quantum memory in the following way : For two tuples $(p_1, ..., p_m)$ and $(q_1, ..., q_m)$ both of which represent qualified sets and satisfies the condition $q_i \geq p_i \; \forall \; i \; (1 \leq i \leq m)$, the tuple $(q_1, ..., q_m)$ can be replaced by the tuple $(p_1, ..., p_m)$.

6.3.2 Uniform multipartite access structures

Here we have the condition that every qualified subset comprises at least t participants from no less than k groups. As in the previous constructions : Let the disjoint groups be $P_1, ..., P_m$. Alice maintains tuples $(p_1, ..., p_m)$ where $0 \leq p_i \leq |P_m|$. A set of participants is qualified if the corresponding tuple satisfies

$$\Sigma_{i=1}^{m} p_i \geq t$$

and

$$|\{P_i : (1 \leq i \leq m) \& p_i > 0\}| \geq k.$$

The secret sharing and the reconstruction algorithm is as follows :

Algorithm 4a : [Sharing secret $|S\rangle$]

1. *For every tuple* $T = (p_1, \ldots, p_m)$ *do steps 2 TO 7.*

2. *Count the number of* p_i*'s such that* $p_i > 0$*. Denote this number by* C*.*

3. *If* $(\Sigma_{i=1}^{m} p_i < t)$ *OR* $(C < k)$ *share random states among the participants of the groups.*

4. *If* $(\Sigma_{i=1}^{m} p_i \geq t)$ *AND* $(C \geq k)$ *do steps 5 to 7 .*

5. *Prepare states* $|S_1\rangle, ..., |S_C\rangle$ *from* $|S\rangle$ *via a* (C, C)*-QTSS.*

6. *For each* p_i *such that* $p_i > 0$*, share one unassigned state* $|S_j\rangle$ *in the group* P_i *via a* $(p_i, |P_i|)$*-QTSS.*

7. *For each* p_i *such that* $p_i = 0$*, share random states among the participants of the group* P_i*.*

8. STOP

Algorithm 4b : [Secret Reconstruction from a qualified set Q]

1. *Recall the corresponding tuple* $(p_1, ..., p_m)$*.*

2. *Note the groups from which positive number of participants arrive i.e.,* $p_i > 0$*.*

3. *From the noted groups reconstruct the shares* $|S_1\rangle, ..., |S_C\rangle$ *via the reconstruction procedure of the corresponding* $(p_i, |P_i|)$*-QTSS.*

4. *Reconstruct the secret from* $|S_1\rangle, ..., |S_C\rangle$ *via the reconstruction procedure of the* (C, C)*-QTSS.*

5. STOP.

• **Correctness:** Alice checks the conditions for a qualified set and shares the secret in two stages as in the construction for weighted threshold access structures. Correctness is evident from the algorithms 4a and 4b.

• **Privacy:** A forbidden set in this access structure is a set for which the corresponding tuple satisfies ($\Sigma_{i=1}^{m} p_i < t$) OR ($C < k$).Clearly the complement of the collection of forbidden sets is monotone and by the step (iii) of algorithm 4a, we have privacy for our scheme.

Theorem 3: Algorithms 4a and 4b construct a quantum secret sharing scheme for uniform multipartite access structures with secret $|S\rangle$ on groups $P_1, ..., P_m$ with participant threshold t and group threshold k.

• *Remarks* : Again it is possible to reduce the number of tuples to be maintained as in the previous constructions.

6.3.3 Partially hierarchical access structures

Hierarchical relationships among groups can be seen as a star-like partial order, that is, there exists a higher level group on the center node of a star, and the other groups are placed on the leaf nodes. Let the disjoint groups be $P_1, ..., P_m$ with P_1 being the central higher level group. The conditions for being a qualified set are as follows :

1. Participants in P_1 satisfying a threshold t_1' can reconstruct the secret.
2. Atleast t_1 (but less than t_1') participants from P_1 combining with atleast t participants from no less than k of the remaining groups can reconstruct the secret.

Alice maintains tuples $(p_1, ..., p_m)$ where $0 \leq p_i \leq |P_m|$. The secret sharing and the reconstruction algorithm is as follows :

Algorithm 5a : [Sharing secret $|S\rangle$]

1. *For every tuple* $T = (p_1, \ldots, p_m)$ *do steps 2 TO 9.*

2. *If* $p_i \geq t'_1$, *share* $|S\rangle$ *in group* P_1 *via a* $(t'_1, |P_1|)$*-QTSS and share random states to the remaining participants.*

3. *Count the number of* p_i*'s such that* $p_i > 0$ $(2 \leq i \leq m)$. *Denote this number by* C.

4. *If* $(t_1 \leq p_1 < t'_1)$ *AND* $(\Sigma_{i=2}^{m} p_i \geq t)$ *AND* $(C \geq k)$ *do steps 5 to 9.*

5. *Prepare states* $|S_1\rangle, |S_2\rangle$ *from* $|S\rangle$ *via a* $(2, 2)$*-QTSS.*

6. *Share* $|S_1\rangle$ *in group* P_1 *via a* $(t_1, |P_1|)$*-QTSS.*

7. *Prepare states* $|S_2^2\rangle, ..., |S_C^2\rangle$ *from* $|S_2\rangle$ *via a* (C, C)*-QTSS.*

8. *For each* p_i $(i \geq 2)$ *such that* $p_i > 0$, *share one unassigned state* $\left|S_2^j\right\rangle$ *in the group* P_i *via a* $(p_i, |P_i|)$*-QTSS.*

9. *For the remaining* p_i *such that* $p_i = 0$, *share random states among the participants of the group* P_i.

10. STOP

Algorithm 5b : [Secret Reconstruction from a qualified set Q]

1. *Recall the corresponding tuple* $(p_1, ..., p_m)$.

2. *If* $p_1 \geq t'_1$, *reconstruct* $|S\rangle$ *by the reconstruction procedure of the* $(t'_1, |P_1|)$*-QTSS.*

3. *If* $t_1 \leq p_1 < t'_1$ *do steps 4 to 8.*

4. *Note the groups from which positive number of participants arrive i.e.,* $p_i > 0$ $(i \geq 2)$.

5. *From the noted groups reconstruct the shares* $|S_2^2\rangle, ..., |S_C^2\rangle$ *via the reconstruction procedure of the corresponding* $(p_i, |P_i|)$*-QTSS.*

6. *Reconstruct the share* $|S_2\rangle$ *from* $|S_2^2\rangle, ..., |S_C^2\rangle$ *via the reconstruction procedure of the* (C, C)*-QTSS.*

7. *Reconstruct* $|S_1\rangle$ *from the group* P_1 *by the reconstruction procedure of the* $(t_1', |P_1|)$*-QTSS.*

8. *From* $|S_1\rangle$ *and* $|S_2\rangle$*, reconstruct the secret* $|S\rangle$ *by the reconstruction procedure of the* $(2, 2)$*-QTSS.*

9. STOP.

• **Correctness and Privacy :** Alice checks the conditions for a qualified set and shares the secret in three stages. First she shares states to the participants of the highest level P_1 according to the threshold t_1. Then Alice checks the remaining conditions for a qualified set. From the secret $|S\rangle$, Alice prepares two states $|S_1\rangle, |S_2\rangle$ via a $(2, 2)$-QTSS. Finally Alice shares $|S_1\rangle$ in the group P_1 and $|S_2\rangle$ in the remaining groups as per the threshold conditions. Secret reconstruction and privacy are evident from the algorithms 5a and 5b.

Theorem 4: Algorithms 5a and 5b construct a quantum secret sharing scheme for partially hierarchical access structures with secret $|S\rangle$ on groups $P_1, ..., P_m$ with P_1, the highest level group with threshold t_i'. From the remaining groups t participants from

no less than k groups combining with t_i participants of group P_1 can reconstruct the secret.

6.3.4 Quasi-threshold multipartite access structures

Each group P_i has a threshold t_i $(1 \leq i \leq m)$, and a special participant p is contained in some group, w.l.o.g., P_1. There are two cases such that the secret can be reconstructed, that is, a set of at least t_1 participants from $P_1 \backslash \{p\}$ is qualified; otherwise, a set contains at least $(t_1 - 1)$ participants from $P_1 \backslash \{p\}$ and no less than k $(1 \leq k \leq m - 1)$ other groups that achieve their thresholds are also qualified.

This scheme can be constructed by a combination of above-mentioned partially hierarchical, uniform multipartite and the uniform threshold access structures. We omit the exact construction to avoid unnecessary repetitions.

6.3.5 Security

The security of our scheme depends on the underlying quantum threshold scheme being used. For (t, n) quantum threshold secret sharing schemes in the literature [128, 131] the authors have shown security against popular quantum attacks such as intercept-and-resend attack, entangle-and-measure attack, man-in-the-middle attack and trojan horse attack. Since our construction uses such (t, n) quantum schemes as a building block, our schemes inherit security against such attacks. In addition to this the use of error-correcting codes gives certain resilience to our scheme. If the distance of the error-correcting code used is d, then even if d shares are tampered, the secret can be correctly recovered.

6.4 Comparison

In this section we draw comparisons of our construction with some recent constructions of hierarchical information splitting schemes. One advantage of our work is the simplicity of the constructions. Another advantage of our construction is the ability to handle arbitrary number of groups and participants. For example in [153] the authors mention that a more general hierarchical quantum information splitting scheme should involve more than three parties and it will be much more complicated and cannot be obtained by directly generalizing their scheme. Clearly our constructed scheme overcomes this. In [154] a multiparty hierarchical quantum information splitting scheme is proposed, where the agents are divided into two grades (G1 and G2) and the number of agents in both grades can be arbitrary in principle. The agents of grade G1 have a larger authority (or power) than the ones of grade G2 to recover the sender's secret state. In comparison to this construction our construction can handle arbitrary number of groups of participants. Again the scheme of [160] constructs a hierarchical scheme for a specific number of parties. In [130] the authors use different classical bits strings as secrets of different levels, while in our constructions(Algorithms 1,2),we need only a single quantum secret $|S\rangle$ and different updates of this secret is shared among different levels. Since classical operations are not used we need to use trap codes and random permutations for the ✠ operations. Also depending on the underlying threshold secret sharing schemes being used we can share any level quantum secret in the hierarchical scheme. So in every sense our scheme is a general hierarchical scheme. Finally another notable advantage of our schemes lie in their flexibility. We have seen the main construction can be very easily modified to construct quantum schemes for other access

structures. We cover a lot of access structures and to our best knowledge, quantum schemes for some of the access structures have not been constructed previously.

Chapter 7

Embedding Hard Functions in Secret Sharing Schemes

7.1 Introduction

In this chapter we construct secret sharing schemes which are robust against an external adversary who can observe and change a fraction of the share strings. The ideas and techniques of this paper can be applied to the constructed schemes of chapters 3 and 4 and in a variant of *secure broadcasting* in the presence of external adversary with a certain constraint, i.e., the adversary can only see and modify some fraction of the communicated messages. The main technique utilized in this chapter is to embed functions with known bounds on their *randomized decision tree complexity* in the share strings.

In the context of quantum secret sharing we consider the resilience (defined below) of our schemes in chapters 5 and 6 against an adversary/ eavesdropper who can query and modify a bounded number of the states. This is done by embedding decoy states which are outputs of functions with known lower bounds on their *quantum query complexity*. Quantum query complexity is the optimal number of queries that a quantum algorithm makes to compute a function. The use of quantum query complexity of Boolean

functions in this context is new to our work.

7.1.1 Our Contribution

1. Achieving resilience of the constructed evolving scheme using functions with bounded *quantum query complexity*. Utilizing quantum query complexity is advantageous in reducing share size against bounded leakage as compared to similar existing ones.

2. To reduce share size as compared to that of [47], we propose a new technique to embed a function with known randomized decision tree complexity in the string generated by the method of [47]. We see that the scheme is robust against an adversary who can see and modify a fraction of the share strings. All the operations in our construction are AC^0 implementable.

3. We apply the above technique to make a fine-grained analysis of secret sharing schemes i.e., analysis of share sizes when the adversary is moderately powerful (computationally bounded). As mentioned before we discuss the applicability of our technique in the context of secure broadcasting.

We achieve security even if the function whose output we are embedding in the share string is known to the adversary.

7.2 Resilience from lower bounds on quantum query algorithms

For a quantum secret $|S\rangle$, let $|Y_S\rangle$ denote a share. Consider an adversary who can make d quantum queries to get a state $|Y_d^1\rangle$ and/or can also modify d shares to transform $|Y\rangle$ to $|Y_d^2\rangle$. The following two items are generally considered.

1. $|Y_d^1\rangle$ is independent of the secret $|S\rangle$.
2. $Pr[Rec(|Y_d^i\rangle = |S\rangle)] \geq 1 - \eta$ for some $\eta \in (0, 1)$, $\forall i = \{1, 2\}$.

Item 2 is generally called resilience/ robustness. To make resilient schemes one idea is to share an encoding of the secret by a QECC instead of sharing the secret [113]. Since we use CSS codes in our construction, it already gives our scheme certain resilient/ robustness capabilities. Our main focus is to achieve item 1 by utilizing lower bounds on quantum query complexity of certain functions.

• **Quantum Query Complexity.** Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, suppose $x = x_1x_2 \ldots x_n \in \{0,1\}^n$ is an input of f (x_i denotes i-th bit). A *quantum query algorithm* for f computes $f(x)$, given queries to the bits of x. Quantum queries are made to an oracle which is defined as $O_x |i, b\rangle = |i, b \oplus x_i\rangle$. A T-query quantum algorithms is a sequence of unitaries $U_T O_x U_{T-1} O_x ... O_x U_0$ where U_i's are fixed unitaries and O_x depends on x. The algorithm can be described as starting from a fixed state $|\phi_0\rangle$, it performs a sequence of unitaries as mentioned previously to obtain the final state $|\phi_x\rangle = U_T O_x U_{T-1} O_x ... O_x U_0 |\phi_0\rangle$. The state $|\phi_x\rangle$ is measured with a $0-1$ positive operator-valued measurement and the measurement result is regarded as the output of

the algorithm. Let $m(x)$ denotes the measurement result of $|\phi_x\rangle$. If $\forall x$,

$$Pr[m(x) = f(x)] \geq 1 - \epsilon,$$

where $\epsilon < \frac{1}{2}$, then the quantum query algorithm is said to compute $f(x)$ with bounded error ϵ. If the error $\epsilon = 0$, then $f(x)$ is computed exactly. The *quantum query complexity* of a function f is the number of queries that an optimal quantum algorithm should make in the worst case to compute f. It is denoted by $Q_\epsilon(f)$ and in the exact setting it is denoted by $Q_E(f)$. For more on quantum query complexity we refer the reader to [5, 4, 6, 45, 105, 134].

• **Achieving Resilience.** To make a scheme resilient against quantum queries, we perform the following operations to the shares $\left|r_{(n_1,\ldots,n_g)}\right\rangle$. In short we denote the state as $|r\rangle$. For the following algorithms recall the **Algorithm 1** of chapter 5.

• **Algorithm 3** : Embed a function

1. *On the state $|r\rangle$ obtained in Step 3 of **Procedure**: RandomShareGen, apply a function with a known lower bound on the quantum query complexity to get the state $|r\rangle\,|f(r)\rangle$.*
2. *Embed $|f(r)\rangle$ as decoy states in random positions in $|r\rangle$ to get the state $|r_f\rangle$.*
3. *On $|r_f\rangle$ the random permutation σ_1 and proceed as in **Procedure**:RandomShareGen.*
4. STOP.

To remove the embedded function outputs/decoy states we do the following algorithm

• **Algorithm 4** : Remove decoy states

1. *Apply inverse permutation* σ_1^{-1} *to get* $|r_f\rangle$.
2. Locate and collect decoy states in $|r_f\rangle$ to get the state $|r\rangle\,|f_r\rangle$.
3. STOP.

• **Correctness of algorithms 3 and 4.** The idea is to take the state $|r\rangle$, apply a Boolean function to it and embed the output as decoy states at random positions inside $|r\rangle$. To see that the above technique works in this case, we first ensure that the number of queries the adversary makes is less than the lower bound on the quantum query complexity of the embedded function being used. In a recent work of Bun et al. [34], the authors showed that for "k-distinctness function" lower bound on the quantum query complexity is $\Omega(n^{3/4-1/(2k)})$. We embed the k-distinctness function as f and stipulate that the eavesdropper makes at most $d_{n_t}(< \Omega(n_t^{3/4-1/(2k)}))$ queries at time t. By the definition of quantum query complexity, the adversary cannot compute and locate the functional valued decoy states. Note that after the decoy states are embedded, the state is appended with the secret state and a random permutation is applied followed by a quantum one-time pad. Since the state $|r\rangle$ needs to be separated from the secret state of one generation to recover the secret of the previous generation, without correctly computing the decoy states this is not possible. Hence the adversary/ eavesdropper learns no information about the secret. • The embedded function can be made public and even then the adversary cannot learn any information about the secret, • only a constant number of decoy states is enough to give us security which reduce the overall dimensions of the shares and makes security easier to achieve, • this method also helps to reduce the amount of quantum memory to a considerable extent (due to

the constant number of decoy states used).

7.3 Preliminaries

We shall assume basic definitions on secret sharing and related concepts. The tools needed for our constructions are, secret sharing, AC^0 circuits/complexity class[9], [139], near optimal error-correcting codes [49], statistical distance, k-wise independent generators [23], randomized decision tree complexity [9] and some properties of Boolean functions [33]. We mainly adopt the notations and definitions of [23], [47].

7.3.1 Randomized decision tree complexity

Our main contribution is based on the *randomized decision tree complexity* of a Boolean function. We briefly describe it here.

A randomized decision tree A on n variables is a distribution over all deterministic decision tree algorithms on n variables. Given an input x, the algorithm first samples a deterministic tree $B \in_R A$ uniformly random, and then evaluates $B(x)$. The cost of a randomized algorithm A on input x, denoted in also by $C(A, x)$, is the expected number of input bits queried by A on x. Let P_f^δ be the set of randomized decision tree algorithms computing f with error at most δ. The two-sided bounded error randomized complexity of f with error $\delta \in [0, 1/2)$ is

$$R_\delta(f) = min_{A \in P_f^\delta} max_{x \in \{0,1\}^n} C(A, x).$$

We write $R(f)$ for $R_0(f)$. A function for which bound on the randomized complexity is known is the *recursive majority* function, $3\mathsf{MAJ}_h$.

Let $\mathsf{MAJ}(x)$ be the Boolean majority function. The *recursive majority function* $3\mathsf{MAJ}_h$ is defined recursively on $n = 3^h$ variables, for every $h \geq 0$. For $h \geq 0$, let x be an input of length n and let $x^{(1)}, x^{(2)}, x^{(3)}$ be the first, second, and third $\frac{n}{3}$ variables of x. Then

$$3\mathsf{MAJ}_h(x) = \mathsf{MAJ}(3\mathsf{MAJ}_{h-1}(x^{(1)}), 3\mathsf{MAJ}_{h-1}(x^{(2)}), 3\mathsf{MAJ}_{h-1}(x^{(3)}))$$

In other words, $3\mathsf{MAJ}_h$ is defined by the *read-once* formula on the complete ternary tree T_h of height h in which every internal node is a majority gate. We identify the leaves of T_h from left to right with the integers $1, \ldots, 3^h$. We have a well known result from [112],

Theorem 7.3.1. *(Magniez et al. [112]) For all $\delta \in [0, \frac{1}{2}]$, we have $(\frac{1}{2} - \delta) \cdot 2.57143^h \leq R_\delta(3MAJ_h) \leq (1.007) \cdot 2.64944^h$.*

7.4 Main results and technical details

7.4.1 Connecting Secret sharing & Randomized decision tree complexity

The **Randomized decision tree** complexity of a Boolean function measures the expected number of queries a *randomized decision tree* makes on an input to decide the output of the function. *So if an adversary sees less bits than the randomized complexity of a function then with high probability it cannot determine the output of the function.* With this idea we do the following steps :

- Do steps 1 to 7 of Algorithm 4. For each of the blocks, compute the $3\mathsf{MAJ}_h$ of each of the bits of the block.

- Embed the function outputs of each block in random locations of the respective blocks.

As for Algorithm 4, if the adversary observes a specified fraction of the string, then on an average it observes the same specified fraction of the blocks. By ensuring the fraction to be less than a specified bound, we can ensure that in some blocks the adversary sees less the number of expected bits (the randomized decision tree complexity of the function) required to compute/decide the output of the embedded function. Hence it cannot decide which bit is the output bit. Since the function bits go into the $\mathsf{ShareGen}_C$ computation, it forms a part of the share. By our construction, the secret can be recovered only when the function outputs are deleted. We can thereby conclude that since the adversary cannot differentiate the share bits from the function output bits, it cannot learn any information about the secret even if the embedded function is known to the adversary.

$3\mathsf{MAJ}_h$ can be computed by AC^0 circuits. Using this function removes the necessity to compute the random permutations which in turn reduces the size of the share string as we no longer have to store all the indices multiple times. Just storing the random locations of the function output is enough for our purpose. If the adversary modifies some of the shares, we can use asymptotically good error correcting codes to ensure robustness.

7.4.2 Notation

We recall the notations of chapter 3. For any $n, k, m \in \mathbb{N}$ with $k, m \leq n$, alphabets Σ_0, Σ, let $(\mathsf{ShareGen}, \mathsf{Reconst})$ be an (n, k) secret sharing scheme with share alphabet Σ_0, message alphabet Σ, message length m. In our case we use Boolean alphabets. Let $(\mathsf{ShareGen}_C, \mathsf{Reconst}_C)$ be an (n_C, k_C) secret sharing scheme from Lemma 3.13 of [47] with alphabet Σ, message length m_C, where $m_C = \delta_0 n_C$, $k_C = \delta_1 n_C$ and $n_C = O(\log n)$ for some constants δ_0 and δ_1. For any constant $a \geq 1$, $\gamma \in (0, 1]$, the paper by Cheng et al. [47] constructs the following $(n_1 = O(n^a), k_1 = \Omega(n_1))$ secret sharing scheme $(\mathsf{ShareGen}_1, \mathsf{Reconst}_1)$ with share alphabet $\Sigma \times [n_1]$, message alphabet Σ, message length $m_1 = \Omega(n_1)$. For clarity we include this as Algorithm 4 as described in [47].

7.4.3 Algorithms, Proofs and Parameters

We shall suitably modify Algorithm 4 to embed "hard" outputs. For brevity we denote *share* by share. The reconstruction algorithm after embedding $3\mathsf{MAJ}_h$ is formalized in Algorithm 6. The correctness of our constructions is mentioned among theorems 7.4.2, 7.4.3 and 7.4.4.

Theorem 7.4.1. *The algorithms 5 and 6 can be implemented by AC^0 circuits.*

Proof. $3\mathsf{MAJ}_h$ can be computed by AC^0 circuits. Also embedding a bit at a random position in a string can also be implemented by AC^0 circuits. All the remaining operations such as random permutation, $\mathsf{approx.\ majority}$, concatenation, partition etc. are AC^0-implementable. The error-correcting codes used in the constructions are also in AC^0. Hence the sharing the secret and the reconstruction in algorithms 4,5 and 6 can

Algorithm 4 Basic Construction of [47]

procedure SHARE GENERATION ($\mathsf{ShareGen}_1$)
$\mathsf{ShareGen}_1 : \Sigma^{m_1} \to (\Sigma \times [n_1])^{n_1}$.
$\bar{n} = \Theta(n^{a-1})$ with large enough constant factor.
$g_\tau : \Sigma_0^{m\bar{n}} \to \Sigma^{m_1}$ is the l-wise independent generator where $l = \Omega(\frac{m\bar{n}log|\Sigma_0|}{log|\Sigma|})^{1-\gamma}$.
For secret $x \in \Sigma^{m_1}$, draw a string $r = (r_1, ..., r_{\bar{n}})$ uniformly from $\Sigma_0^{m\bar{n}}$.
$y = (y_s, y_g)$, *where* $y_s = (Share(r_1), ..., Share(r_n)) \in (\Sigma^n)^{\bar{n}}$ *and* $y_g = g_\tau(r) \oplus x \in \Sigma^{m_1}$.
Get $\hat{y}_s \in (\Sigma^{m_C})^{n_s}$ *from* y_s *by parsing* $y_{s,i}$ *to be blocks each having length* m_C *for every* $i \in [\bar{n}]$, *where* $n_s = \lceil \frac{n}{m_C} \rceil \bar{n}$.
Get $\hat{y}_g \in (\Sigma^{m_C})^{n_g}$ *from* y_g *by parsing* y_g *to be blocks each having length* m_C, *where* $n_g = \lceil \frac{m1}{m_C} \rceil$.
Compute $(Share_C(\hat{y}_{s,1}), .., Share_C(\hat{y}_{s,n_s}),\ Share_C(\hat{y}_{g,1}), .., Share_C(\hat{y}_{g,n_g}))$ *and parse it to be* $y1 = (y1_1, ..., y1_{n_1})$, *where* $n_1 = (n_s + n_g)n_C$.
(Generate a random permutation) $\pi : [n_1] \to [n_1]$ *apply it on* $y1$ *and this is the output.*

Algorithm 5 Embedding $3\mathsf{MAJ}_h$

procedure EMBEDDING $3\mathrm{MAJ}_h$
Do steps $1 - 7$ of Algorithm 4.
Find the greatest integers h_i & $h_j$such that $3^{h_i} \leq |\hat{y}_{s,i}|$ & $3^{h_j} \leq |\hat{y}_{s,j}|$ for all i, j.
Truncate the strings $\hat{y}_{s,i}$ and $\hat{y}_{s,j}$ to lengths 3^{h_i} and 3^{h_j} respectively.
On the truncated strings compute $3\mathsf{MAJ}_{h_i}(\hat{y}_{s,i})$ and $3\mathsf{MAJ}_{h_j}(\hat{y}_{g,j})$ for all i and j by recursively computing $3\mathsf{MAJ}_h(x) = \mathsf{MAJ}\ (3\mathsf{MAJ}_{h-1}(x^{(1)}), 3\mathsf{MAJ}_{h-1}(x^{(2)}), 3\mathsf{MAJ}_{h-1}(x^{(3)}))$ and storing the intermediate results.
Embed the computed $3\mathsf{MAJ}_h$'s to any random position inside each block of the string obtained after step 2.
On the string obtained in step 6 compute: $(\mathsf{ShareGen}_C(\hat{y}_{s,1}), \ldots\ldots, \mathsf{ShareGen}_C(\hat{y}_{s,n_s}),\ \mathsf{ShareGen}_C(\hat{y}_{g,1}), \ldots, \mathsf{ShareGen}_C(\hat{y}_{g,n_g}))$ and parse it to be $y1 = (y1_1, \ldots, y1_{n_1})$, where $n_1 = (n_s + n_g)n_C$.
$y1$ is the output.

Algorithm 6 Reconstruction

1: **procedure** RECONSTRUCTION
2: Compute $\mathsf{Reconst}_C$.
3: Retrieve all the functional values from y_1.
4: Delete all the functional values from y_1. Denote the resulting string also by $y1$.
5: Get y_s and y_g.
6: Apply $\mathsf{Reconst}$ on every entry of y_s to get r.
7: Output $g_\tau(r) \oplus y_g$.

be implemented by AC^0 circuits.

Theorem 7.4.2 (Robustness by embedding $3MAJ_h$)**.** *An adversary which sees a fraction ($\leq 2.57^h/2.3^h$) of the share string cannot compute the $3\mathsf{MAJ}_h$ bits in the string.*

Proof. From Theorem 7.3.1, $R(3\mathsf{MAJ}_h) \geq \frac{2.57^h}{2}$. Theorem 7.3.1 implies that the fraction of the string that the adversary has to observe in order to correctly compute is greater than $2.57^h/2.3^h$. Let us denote this fraction by f and suppose that the fraction of the string the adversary sees is less than f. This means that on an average the number queries that the adversary makes in each block is less than f which is the expected number of bits needed to be queried to compute the $3\mathsf{MAJ}_h$ function. Since the function output bits are embedded in random positions inside the block, the adversary cannot compute the function bits. Hence it cannot differentiate between the function output bits and the other bits. Since the function bits go into the $Share_C$ computation, without identifying and deleting the function bits, the adversary cannot learn any information about the secret. This makes the construction robust against an adversary observing a fraction of the string. If the adversary modifies a fraction of the shares then we can encode the outer block via asymptotically good error-correcting codes and the secret can be reconstructed by the properties of the error-correcting codes being used.

Theorem 7.4.3. *Algorithms 4, 5 and 6 generate an n-out-of-n-secret sharing scheme i.e., n parties have to participate to recover the secret. The privacy threshold of the secret sharing scheme thus obtained is* $\Omega(n)$.

Proof. The changes that we make to algorithm 1 are done by embedding outputs of certain functions in the blocks. The function inputs are the elements of the blocks itself. Hence our modifications do not change the privacy threshold of the scheme constructed by algorithm 1. Hence we construct an n-out-of-n secret sharing scheme with privacy threshold $\Omega(n)$. However, the share size in our case is considerably lesser.

Theorem 7.4.4. *The share size obtained by embedding the functions is less than that obtained by using random permutations.*

Proof. To keep constructions in AC^0 in Algorithm 4 and to reconstruct the secret, one needs approx. majority to recover indices from repeated bits. *Approximate majority* is required since we need to store the indices multiple times due to the random permutation. In our construction we do not use random permutation which in turn removes the requirement to store multiple indices and compute approx. majority. Hence share size is reduced to a considerable extent in our constructions.

7.4.4 Applications

A fine-grained analysis

The above technique can be utilized to do a fine-grained analysis of share sizes when the adversary is computationally bounded. If we embed Parity in the share string, we can achieve robustness against an AC^0-powerful adversary since Parity cannot be computed

by AC^0 circuits. On the other hand if we embed MAJ (majority cannot be computed by $AC^0[p]$), we can attain robustness against $AC^0[p]$ adversary. But while Parity can be computed in small blocks (since XOR is associative), MAJ cannot be computed in small blocks. Hence we need more storage or embedding bits in the share string to compute MAJ. This implies an increase in the share size in the case of an $AC^0[p]$ adversary as compared to an AC^0 adversary. It is known that $AC^0 \subsetneq AC^0[p]$ as complexity classes. This increase in the share size gives us a fine-grained analysis. We can similarly observe increase in the share size while going higher up in the complexity ladder. For more on complexity classes, we refer the reader to [9].

Secure broadcasting

In this model n parties have local inputs and they share a secret key. Then they communicate over a public broadcast channel and finally each party computes a local output, An external adversary upon seeing a fraction of the message cannot learn any information about the secret. Also for any input, if the adversary corrupts a fraction of the messages, the parties can reconstruct the secret with high probability. Assuming that each party has access to local random bits, one can use secret sharing to the input and broadcast the shares. Now if the adversary is allowed to see and corrupt more than $1/n$ of the messages, then the adversary may choose to observe the share of only one party and the learn the input of that party. One way is to use random permutation on the inputs so that the adversary does not know which message is from which party. But as we have observed before that random permutation increases share size. In our construction we have embedded outputs of functions with high randomized decision tree complexity (high lower bound) in the inputs. Assuming an upper bound on the

fraction of the string that the adversary can observe and modify, we can ensure by embedding function outputs constant times in random positions in each share, that the adversary sees less number of bits that is required to identify the function output. Hence as compared to the random permutation construction of [47], our construction reduces share size and hence the communication complexity.

Chapter 8

Generalized Matroid Ports and Non-Perfect Secret Sharing

8.1 Introduction

Results In this chapter we make the following contributions :

1. A forbidden minor characterization for generalized matroid ports.
2. Applications of generalized ports.
3. A connection between multipartite quasi-matroids, multipartite generalized ports and the associated integer polymatroid.

8.2 Preliminary Definitions and Results

In this chapter we shall use the definitions and notations as in [66] and [69].

• A *polymatroid* is a pair $S = (Q, f)$ formed by a finite ground set Q and a rank function $f : P(Q) \to \mathbb{R}$ satisfying the following properties.

- $f(\phi) = 0$.

- f is monotone increasing: if $X \subseteq Y \subseteq Q$, then $f(X) \leq f(Y)$.
- f is submodular: $f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)$ for every $X, Y \subseteq Q$.

• A polymatroid is called integer if its rank function is integer-valued.

• A matroid $M = (Q, r)$ is an integer polymatroid such that $r(\{x\}) \in \{0, 1\}$ for every $x \in Q$.

• **Notation :** For subsets X, Y of Q, $f(X|Y) := f(X \cup Y) - f(Y)$.

• For a polymatroid $S = (Q, f)$ and a set $Z \subseteq Q$, the polymatroids $S\backslash Z = (Q - Z, f_{\backslash Z})$ and $S/Z = (Q - Z, f_{/Z})$ are defined, respectively, by $f_{\backslash Z}(X) = f(X)$ and $f_{/Z}(X) = f(X|Z)$. These are respectively known as deletion and contraction.

• Every polymatroid that can be obtained from S by repeatedly applying these operations is called a minor of S. Every minor of S is of the form $(S\backslash Z_1)/Z_2$ for some disjoint sets $Z_1, Z_2 \subseteq Q$.

• The minors of a matroid are also matroids. The dual of a matroid $M = (Q, r)$ is the matroid $M^* = (Q, r^*)$ with $r^*(X) = |X| - r(Q) + r(Q - X)$ for every $X \subseteq Q$. We have $M^{**} = M$. In addition, $(M\backslash Z)^* = M^*/Z$ and $(M/Z)^* = M^*\backslash Z$ for every $Z \subseteq Q$.

Let P be a finite set of participants, $p_0 \notin P$ a special participant called the dealer, and $Q = P \cup \{p_0\}$.

Definition 12. *Matroid ports : Let $S = (Q, f)$ be a polymatroid with $f(p_0) > 0$ and $f(\{p_0\}|P) = 0$. The access structure $\Gamma_{p_0}(S) = (\mathcal{A}, \mathcal{B})$ on P is defined by*

- $\mathcal{A} = \{A \subseteq P : f(\{p_0\}|A) = f(\{p_0\})\}$,
- $\mathcal{B} = \{B \subseteq P : f(\{p_0\}|B) = 0\}$.

This is an access structure. If M is a matroid, then the access structure $\Gamma_{p_0}(M)$ is perfect and it is called the port of the matroid M at the point p_0.

Definition 13. *Quasi-matroid[69]: A quasi-matroid is an integer polymatroid $S = (Q, f)$ such that there exists $p_0 \in Q$ with $f(\{x\}) = 1$ for every $x \in Q - \{p_0\}$.*

Definition 14. *A matroid $\mathcal{M} = (P \cup P_0, r)$ is said to be P_0 uniform [69] if $r(P_0) = |P_0|$ and $r(X) = min\{r(X \cup P_0), r(X - P_0) + |X \cap P_0|\}$.*

Now we are in a position to define generalized matroid ports.

Definition 15. *Generalized matroid ports:[69] Let P and P_0 be disjoint finite sets and $M = (P \cup P_0, r)$ be a matroid such that $r(P_0) = |P_0|$ and $r(P_0|P) = 0$. Then the generalized port of the matroid M at the set P_0 is the access structure $\Gamma_{P_0}(M) = (\mathcal{A}, \mathcal{B})$ defined by*

- $\mathcal{A} = \{A \subseteq P : r(P_0|A) = r(P_0)\}$,
- $\mathcal{B} = \{B \subseteq P : r(P_0|B) = 0\}$.

Here we mention some properties of generalized matroid ports which will be required later[66].

- The dual of a generalized matroid port is a generalized matroid port.
- If $\Gamma = (\mathcal{A}, \mathcal{B})$ is a connected generalized matroid port, then there exists a unique quasi-matroid S such that $\Gamma = \Gamma_{p_0}(S)$.

Seymour gave a forbidden minor characterization of matroid ports [138]. In that paper he used an alternate definition of matroid ports in terms of circuits given by Lehman [107] and blockers and clutters which we discuss briefly.

Definition 16. *Matroid port:[138] If Ω is an element of a connected matroid M, then the collection $\{C - \{\Omega\} : \Omega \in C \in \mathcal{C}\}$ where $\mathcal{C}$ is the collection of circuits of M is called a matroid port.*

A matroid is said to be connected if for every two distinct points there exists a circuit containing those two ports. Clearly, all ports of a connected matroid are connected. A connected matroid is determined by the circuits that contain some given point. Therefore, if Γ is a connected matroid port, there exists a unique connected matroid M with $\Gamma = \Gamma_{p_0}(M)$.

Definition 17. *Clutter:[138] A clutter L is a collection of sets such that for $A_1, A_2 \in L, A_1 \not\subset A_2$.*

Definition 18. *Blocker:[138] The blocker of a clutter L, denoted by $b(L)$ is the collection of the minimal sets which have non-empty intersection with each member of L.*

Here we note that :-

- $b(L)$ is a clutter.
- $b(b(L)) = L$.

Matroids can be described in terms of clutters and blockers as follows : Let V be the set of vertices of L. For any set Z, $L\backslash Z$ is defined as $\{A \in L : A \cap Z = \Phi\}$, and L/Z is the collection of minimal members of $\{A - Z : A \in L\}$. Then similar to the previous definition it can be verified that $b(L\backslash Z) = b(L)/Z$, and $b(L/Z) = b(L)\backslash Z$. If Z_1 and Z_2 are disjoint sets, then $(L\backslash Z_l)/Z_2 = (L/Z_2)\backslash Z_1$. A *minor* of L is a clutter which

may be obtained from L by repeated use of the $\backslash, /$ operations. Any minor L' of L is expressible in the form $(L\backslash Z_1)/Z_2$, where Z_1, Z_2 are disjoint and $Z_1 \cup Z_2 \cup (V') = V$.

The two definitions of matroid ports seen so far are equivalent due to the following fact relating rank and circuits of a matroid :-

Lemma 8.2.1. *Suppose $M = (Q, r)$ is a matroid with rank function r, then a subset $X \subseteq Q$ is a circuit iff X is non-empty and for all $x \in X$, $r(X - x) = |X| - 1 = r(X)$.*

8.2.1 Minimal non-ports

Here we state the forbidden minor characterization of matroid ports due to Seymour [138].

- Minor-minimal non-ports are $P_4, Q_4, b(Q_4)$ and $J_s, s \geq 3$. where
- $P_4 = \{\{1,2\},\{2,3\},\{3,4\}\}$,
- $Q_4 = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\}\}$,
- $b(Q_4) = \{\{1,2\},\{1,3\},\{2,3,4\}\}$ and
- $J_s = \{\{1,...,s\}\} \cup \{\{0,i\} : i = 1,...,s\} (s \geq 3)$.

Definition 19. *If L is a clutter and $A_1, A_2 \in L$ are distinct, $L(A_1, A_2)$ is defined as $\cap\{A \in L : A \subseteq A_1 \cup A_2\}$.*

The following theorem was proved in [138],

Theorem 8.2.2. *If L is a clutter, the following are equivalent:*

1. *L is a port.*

2. *For $A_1, A_2, A_3 \in L$ and $x \in A_1 \cup A_3 - A_2$, there exists $A \in L$ with $A \subseteq (A_3 \cup (A_1 \cup A_2 - L(A_1, A_2))) - \{x\}$.*

3. *There do not exist $A_1, A_2 \in L$ and $B_1, B_2 \in b(L)$ and distinct $x, y \in UL$ such that $A_1 \cap B_2 = A_2 \cap B_1 = A_2 \cap B_2 = \{y\}$ and $A_1 \cap B_1 = \{x, y\}$.*

4. *L has no minor $P_4, Q_4, b(Q_4)$ or $J_s, (s \geq 3)$.*

8.3 Results

$\Gamma_{P_0}(M)$ is a clutter and clutters of this form are called generalized matroid ports. From a previous observation that given a generalized matroid port there exists a quasi-matroid whose port(corresponding to the element of rank different from 1) is equal to the given generalized matroid port, we can conclude that the class of generalized ports is closed under minors. We shall define a k-port where $|P_0| = k$. It should be noted that when considering minors of ports a k-port can contain a $k-1$-port or $k-2$-port and so on as minors but not vice-versa. Hence in this way it makes sense to consider the problem of finding minors which are not generalized matroid ports.

8.3.1 Generalized non-ports

• **Note :** For our results we shall only consider graphic matroids.

From the above discussion we can now classify the generalized non-ports as follows : Since matroid ports are also generalized matroid ports where the set P_0 has only one

element the generalized non-ports are constructed by building up from $P_4, Q_4, b(Q_4)$ and $J_s(s \geq 3)$. We have noted before that a quasi-matroid is an integer polymatroid where all but one element has rank one. The remaining element can be thought of as an element with more weight. This motivates that when constructing generalized non-ports, a set of elements are to be deleted. When the special element in the corresponding quasi-matroid has rank one, it is a matroid and hence the non-ports are also generalized non-ports.

Definition 20. *A generalized k-port or a k-port is a generalized matroid port with* $|P_0| = k$.

First we shall characterize 2-non ports and 3-non ports.

8.3.2 2-non ports

Theorem 8.3.1. *The minimal 2-non ports are the following:-*

1. *The clutters which have*

   - $\{\{1,2\},\{2,3\},\{3,4\}\}$,
   - $\{\{1,2\},\{1,3\},\{1,4\},\{2,3\}\}$,
   - $\{\{1,2\},\{1,3\},\{2,3,4\}\}$,

   as minors with number of elements in the ground set 5,

2. *The clutters which have* $\{\{1,...,s\}\} \cup \{\{0,i\} : i = 1,...,s\}(s \geq 3)$ *as minor with* $s+3$ *elements in the ground set.*

3. *or a union of two non-ports as minors with the number of elements in the ground set 6.*

Proof. Suppose P_0 is a set with two points. Since $r(P_0) = |P_0|$, P_0 is an independent set. Let C be a circuit containing P_0. After deleting two elements from C, there can be either of the two cases :

1. It can be a matroid port or can have a matroid port as a minor in which case the number of elements in the ground set must be at least 5. (For example, in a graphic matroid, deleting two adjacent edges from a cycle has the same effect as deleting one edge from the cycle.) In this case the 2-non ports are same as the non-ports.

2. It can have the union of two matroid ports as minor.(For example, in a graphic matroid deleting two non-adjacent edges gives rise to two paths.) Hence the 2-non ports are the disjoint union of two non-ports. After deleting two points from a collection of circuits containing them, for a 2-non-port we must have either the non-ports or a disjoint union of at least two non-ports as minors. The number of elements in the matroid must be greater than or equal to 6.

□

8.3.3 3-non ports

Theorem 8.3.2. *The minimal 3-non ports are :-*

1. *non-ports,*

2. *non-ports as minors with number of elements in the ground set 5.*

3. *union of two non-ports with the number of elements in ground set 6,*

4. *union of three non-ports with the number of elements in ground set 7.*

Proof. As in the case of 2-non-ports, when we delete three elements from a circuit C, there are three cases :

1. It can have a matroid port as a minor. (For example, in a graphic matroid, deleting three adjacent edges from a cycle has the same effect as deleting one edge from the cycle.) In this case the 3-non ports are same as the non-ports.

2. It can have a 2-port as a minor.(For example in a graphic matroid the three edges to be deleted can have two adjacent edges and a non-adjacent third edge) This case is equivalent to the case of 2-non ports.

3. It can have the union of at least three matroid ports as minor. (As before, in a graphic matroid the three edges to be deleted from a cycle can be non-adjacent). Here the 3-non ports are the union of atleast three non-ports.

   Again, deleting three elements from a collection of circuits containing them gives rise to either of the above as minors. Also the number of elements in the matroid must be greater than or equal to 7.

□

8.3.4 k-non ports

k-non ports($k \geq 3$) are characterized as follows :-

Theorem 8.3.3. *The minimal k-non ports are the non-ports,*

1. *non-ports as minors with number of elements in the ground set $k+1$ and*

2. *for every i, $2 \leq i \leq k$, the union of i non-ports with number of elements in the ground set $k+i$.*

Proof. Here we delete k elements from a circuit. Now a set of k elements can be partitioned in $p(k)$ ways, where $p(k)$ is the partition number. (In the language of graphs each partition corresponds to the adjacent edges in a circuit.) So we have $p(k)$ cases. Among the $p(k)$ cases we have three notable ones.

1. The number of partitions is one. This case is equivalent to deleting one element from a circuit. Hence here the minimal k-non ports are the minimal non-ports.

2. The number of partitions is k. (In a graphic matroid we delete k edges where all are non-adjacent.) In this case there are k unions of matroid ports as minors. Hence the minimal k-non ports are the disjoint union of k many non-ports.

3. Let the number of partitions be t with $1 < t < k$. (In the cycle of a graphic matroid, consider the edges in a partition to be adjacent and edges in different partitions to be non-adjacent.) This case reduces to the case of t-non ports and hence the minimal k-non ports are the t disjoint unions of non-ports. The number of elements follow immediately.

□

8.3.5 Applications

Here we point out some potential applications of the forbidden minor characterization. Such a forbidden minor characterization can help in determining if an access structure is a generalized port. For example in [69] , the following access structure was shown not to be a generalized port. Our characterization gives an alternate proof. Further such a characterization can prove to be uselful to find out optimal complexities of various non-perfect secret sharing schemes.

The forbidden minor characterization of matroid ports was applied in using a method called the *independent sequence method* [114] where characterizations of matroid ports were obtained in terms of independent sequences and a complexity parameter of secret sharing schemes. Using the forbidden minor characterization for generalized ports we can hope to get similar results for generalized ports which is a future line of work.

8.4 Generalizing Ideal Multipartite Access Structures To The Non-Perfect Case

In this section we briefly consider multipartite access structures. In multipartite secret sharing participants are distributed into separate classes as in hierarchical schemes. So it makes sense to consider the non-perfect version of such schemes. We shall use the connection between non-perfect schemes, generalized matroid ports and quasi-matroids. Ideal multipartite access structures were considered in [66].

8.4.1 Multipartite Access Structures and Multipartite Matroids

• The $m-partition$ $\Pi = (X_1, ..., Xm)$ of a set X is a partition of the set X into m disjoint subsets, $X = X_1 \cup ... \cup X_m$.

• A permutation σ on X is a $\Pi-permutation$ if the subsets X_i remain invariant under σ, i.e., $\sigma(X_i) = X_i$ for all $i = 1, ..., m$.

• A combinatorial object on X is $m-partite$ if it is $\Pi-partite$ for some $m-partition$ Π.

• A family $\Lambda \subseteq \mathcal{P}(X)$ of subsets of X is $\Pi - partite$ if it remains invariant under the action of σ, i.e., $\sigma(\Lambda) = \{\sigma(A) : A \subseteq \Lambda\} = \Lambda$ for every Π-permutation σ on X.

8.4.2 Quasi-matroids and generalized ports

Given a matroid $M = (P \cup P_0, r)$ such that $r(P_0) = |P_0|$, consider the polymatroid $M|P_0$ with ground set $Q = P \cup \{p_0\}$ and rank function f defined by $f(X) = r(X)$ and $f(X \cup p_0) = r(X \cup P_0)$ for every $X \subseteq P$. We observe that $M|P_0$ is a quasi-matroid. In addition, $(M|P_0)\backslash\{p_0\} = M\backslash P_0$ and $(M|P_0)/p_0 = M/P_0$. We shall make use of the following observation in [66] : Let $S = (Q, f)$ be a quasi-matroid. Then there exists a unique P_0-uniform matroid $M = M(S) = (P \cup P_0, r)$ such that $M(S)|P_0 = S$. Our approach is to make use of this fact : given a multipartite quasi-matroid, we go to the corresponding multipartite P_0 uniform matroid and from there we go to the assiciated integer polymatroid.

Multipartite Quasi-matroids and multipartite generalized ports

The following theorem is clear.

Theorem 8.4.1. *Let M be a quasi-matroid with ground set Q. Consider a point $p_0 \in Q$ and partitions $\Pi = (P_1, ..., P_m)$ and $\Pi_0 = (\{p_0\}, P_1, ..., P_m)$ of the sets $P = Q - \{p_0\}$ and Q, respectively. Then the generalized matroid port $\Gamma_{p_0}(M)$ is Π-partite if and only if the quasi-matroid M is Π_0-partite.*

For definitions in this sections, we refer the reader to [66]. Let $J_m = \{1, 2, ..., m\}$.

Now we make a connection between multipartite quasi-matroids and integer polymatroids.

Theorem 8.4.2. *Let $\Pi = (Q_1, ..., Q_m)$ be an m-partition of a set Q which is the ground set of a Π-partite quasi-matroid $M = (Q, h)$. As in the case of multipartite matroids we have a corresponding integer polymatroid J_m.*

Proof. We know that : Let $\Pi = (Q_1, ..., Q_m)$ be an m-partition of the set Q. Then the corresponding partition of the set $P \cup P_0$ is as follows : Replace the element p_0 in any Q_i by the set P_0. The remaining correspondence follows from the correspondence between multipartite matroids and integer polymatroids.

□

We can also prove that an m-partite quasi-matroid is determined by the rank of the special element(the one with rank greater than 1), its associated integer polymatroid

and the m-partition of the ground set. From the above discussion, given a (multipartite) quasi-matroid, we get a unique (multipartite) P_0-uniform matroid and finally using the correspondence between multipartite matroids and integer polymatroids we get a correspondence between a multipartite quasi-matroid and an integer polymatroid.

Finally we have the following theorem generalizing to generalized martroid ports:

Theorem 8.4.3. *Let $\Pi = (P_1, ..., P_m)$ be a partition of a set P and let Γ be a connected Π-partite access structure on P. Consider $\Delta = supp(\Gamma)$. Then Γ is a generalized matroid port if and only if there exists an integer polymatroid $Z = (J_m, h)$ with $h(\{i\}) \leq |P_i|$ for every $i \in J_m$ such that Δ is compatible with Z and $min\Pi(\Gamma) = min\{u \in \mathcal{B}(Z, X) : X \in \Delta\}$.*

Proof. What we get here is a P_0-uniform matroid. From this matroid we can get the corresponding quasi-matroid. The proof is same as in the case of multipartite matroid ports and is omitted.

□

Multipartite access structures such as multilevel access structures, compartmented access structures have been expressed in terms of matroid and have been proved to be ideal. Our connection helps in expressing the non-perfect version of such schemes in the language of matroids and polymatroids. The actual constructions of non-perfect multipartite schemes is our future line of work.

Chapter 9

A short note on intervals in Hales-Jewett Theorem

One of the main results in conbinatorics and Ramsey theory is the Hales-Jewett Theorem. For $m, n \in \mathbb{N}$, let $[m]^n$ denote the set of all n-letter words with alphabets from $[m] = \{1, 2, \ldots, m\}$. For a word $w \in [m]^n$, $S \subseteq [n]$ and $i \in [m]$, $w(S, i)$ is obtained from w by replacing the jth letter with i for all $j \in S$. A *combinatorial line* in $[m]^n$ is defined as the set of words $\{w(S, 1), w(S, 2), \ldots, w(S, m)\}$ with the *wildcard set* $S \neq \phi$. The Hales-Jewett Theorem [86] says that for $m, r \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that any r-colouring of $[m]^n$ contains a monochromatic combinatorial line. In [53] the authors asked the following question:

• Do there exist $m \geq 4, r \geq 2$ and $c > 1$, $m, r, c \in \mathbb{N}$ such that there are r-colourings of $[m]^n$ containing no monochromatic combinatorial line whose wildcard set is the union of at most cr intervals ?

We answer this question in the positive :

Theorem 9.0.1. *There is a 5-colouring of* $[4]^n$ *containing no combinatorial line whose wildcard set is the union of at most* 25 *intervals* $(c = 5, r = 5)$. *If* $gcd((m-1)(m-2), cr) = 1$, *then the conjecture holds.*

We shall follow the ideas and notations of [53].

Let $t = (t_1, t_2, t_3, t_4) \in (\mathbb{Z}/cr\mathbb{Z})^4$. Let w be a word in $[4]^n$. For the word w, let $T'(w) = \Sigma_{j\in[n]} t_{w(j)}$.

Contract the intervals on which w is a constant to a single letter to get $\bar{w}$. Let w^+ be obtained by ading the letter 1 to the start and end of w. For the colouring $T^+(w) = T(w^+)$.

Now $w(S, i)$ is obtained by replacing the j-th letter in w with i for all $j \in S$.

• **Claim :** For $t_1 = t_3 = t_4 = 3$, $t_2 = -1$, the colouring $T^+ : [4]^n \longrightarrow \mathbb{Z}/cr\mathbb{Z}$ does not contain any combinatorial line which is monochromatic and whose wildcard set is the union of less than cr intervals.

We shall adopt some more notations from [53]. For a combinatorial line $(w(S, 1), ..., w(S, 4))$ with wildcard set S, let $x_i = w(S, i)$. Let S be a disjoint union of q consecutive intervals. Let $w = x_1$. Other than the wildcard set S, the word w^+ consists of a collection of non-empty subwords $w_0, w_1, \ldots, w_q$, where w_{j-1} comes before w_j for all $j = 1, \ldots, q$. Let the starting letter of w_j be f_j and the end letter be l_{j+1}. The authors in [53] show the following :

Theorem 9.0.2. *For any $t_1, t_2, t_3 \in \mathbb{Z}_r$ and $i \in [3]$,*

$$T^+(x_i) = T(x_i^+) = T(w_0) + h_i(l_1, f_1) + T(w_1) + h_i(l_2, f_2) + \ldots + h_i(l_q, f_q) + T(w_q).$$

In our case extending theorem 9.0.2, we have the following

Theorem 9.0.3. *For any $t_1, t_2, t_3, t_4 \in \mathbb{Z}_{cr}$ and $i \in [4]$,*

$$T^+(x_i) = T(x_i^+) = T(w_0) + h_i(l_1, f_1) + T(w_1) + h_i(l_2, f_2) + \ldots + h_i(l_q, f_q) + T(w_q).$$

Proof. Extending the proof of [53] we have the following table which summarizes the values of $h_i(f,l)$ as follows :

(l,f)	$(1,1)$	$(2,2)$	$(3,3)$	$(4,4)$	$(2,3)$	$(3,1)$	$(1,2)$	$(1,4)$	$(2,4)$	$(3,4)$
$h_1(l,f)$	$-t_1$	t_1	t_1	t_1	t_1	0	0	0	t_1	t_1
$h_2(l,f)$	t_2	$-t_2$	t_2	t_2	0	t_2	0	t_2	0	t_2
$h_3(l,f)$	t_3	t_3	$-t_3$	t_3	0	0	t_3	t_3	t_3	0
$h_4(l,f)$	t_4	t_4	t_4	$-t_4$	t_4	t_4	t_4	0	0	0

Since the proof is in the same line as in [53] we omit the proof.

□

Suppose that for $t_1 = t_3 = t_4 = 3, t_2 = -1$, there is a combinatorial line $(w(S,1), ..., w(S,4))$ with wildcard set S, with $T^+(x_1) = T^+(x_2) = T^+(x_3) = T^+(x_4)$ where $x_i = w(S,i)$. The for $i = 1,3,4$, we have,

$$0 = T^+(x_i) - T^+(x_2) = \Sigma_{j=1}^{q}(h_i(l_j,f_j) - h_2(l_j,f_j))$$

Adding these equations we get,

$$0 = T^+(x_1)+T^+(x_3)+T^+(x_4)-3T^+(x_2) = \Sigma_{j=1}^{q}(h_1(l_j,f_j)+h_3(l_j,f_j)+h_4(l_j,f_j)-3h_2(l_j,f_j))$$

It can be verified that for the chosen t_1, t_2, t_3, t_4, for each l, f, we have $h_1(l,f) + h_3(l,f) + h_4(l,f) - 3h_2(l,f) = 6$. Hence

$$0 = T^+(x_1) + T^+(x_3) + T^+(x_4) - 3T^+(x_2) = 6q$$

.

This means that $6q = 0$ in $\mathbb{Z}/cr\mathbb{Z}$. Now set $c = 5$ and $r = 5$. Since $gcd(6, 25) = 1$, we must have $q \geq cr = 25$.

So for $c = 5$, $r = 5$, $m = 4$, there is a 5-colouring of $[4]^n$ containing no monochromatic combinatorial line whose wildcard set is the union of at most $cr = 25$ intervals and the required question is answered in the positive.

Chapter 10

Conclusion

In this chapter we make some concluding remarks about this thesis and discuss some works which are possible extensions of the chapters in this thesis and are currently under preparation as this thesis goes to print. The main focus of this thesis is construction of different variants of secret sharing schemes. We have considered AC^0 implementable dynamic, evolving and perpetual secret sharing schemes. Next we have considered quantum versions of evolving schemes and various multipartite schemes. Finally we have made a note on forbidden minors of generalized matroid ports and discussed possible applications and connections to multipartite structures. The thesis ends with a short note on a question relating to the Hales-Jewett Theorem.

10.1 Chapter 3

In chapter 3 work we proposed two AC^0 implementable secret sharing schemes which can accommodate new parties into the system. First construction is a dynamic scheme where the dealer shares the secret and goes offline after distributing the shares to the parties. Later the parties present in the system redistributes their shares to generate shares of new parties without reconstructing the secret. Second construction is an evolving scheme where the dealer is present throughout and generates shares for the incoming parties with the constraint that the old shares cannot be modified.

There can be scenarios where during the process the secret gets perturbed within a certain Hamming distance. A relevant question is can a scheme be constructed which can handle such a situation without making too many modifications to the shares and without leaking the secret. Another question is related to reproduction number. It is the average number of parties to which a share gets distributed from a single party. We know that when the reproduction number is less than 1, the system eventually ends. Can the share size be reduced in such scenarios is a problem for an ongoing work.

10.2 Chapter 4

This work opens up lots of research directions which we plan to present in an expanded version. Some of these are as follows – utilize data structures to construct AC^0 implementable evolving versions of hierarchical secret sharing schemes and use secret redistribution combined with various combinatorial structures such as hypercubes, simplicial complexes etc. to realize evolving versions of more general access structures with reduced memory usage and keeping the computational complexity of the operations as low as possible.

10.3 Chapter 5

To summarize, in this chapter we initiate the study of a very general class of quantum evolving secret sharing schemes which can handle an unbounded and increasing number of participants. The methods used rely on previously established ones. In the spirit of increasing thresholds, our security technique can handle an increasing number of queries

from the adversary, as long as at any time t the number of queries d_{n_t} the adversary makes is less than the quantum query complexity of the embedded function being used (See Chapter 7). Since only a constant number of decoy states need to be embedded in the share, the share size does not blow up drastically. As a whole this paper mirrors a real-life situation and constructs a resilient quantum secret sharing scheme where the dealer does not know in advance how many participants are going to join in the future. This technique can also be used in the context of secure broadcasting in the presence of external adversary.

One major drawback of this construction is the use of a huge amount of quantum memory. Note that Alice has to maintain/ recall all the tuples. With the increase in generations and participants, these increase at a very high rate. Also the random states need to be generated for every tuple and every generation which requires enhanced computation costs. In the remarks we saw a method to reduce the usage of quantum memory. The method fails when more and more participants arrive and the threshold needs to be updated. So for practical implementation it is necessary to reduce the usage of quantum memory and also to reduce the number of random states. Note that we have not assumed any relation between the participants. Assuming some relations between the participants it might be possible to reduce the usage of quantum memory. Further improvements to this construction is a probable future research direction.

10.4 Chapter 6

The central idea of this chapter is to show how to design a quantum hierarchical secret sharing scheme by maintaining the tuples $(p_1, ..., p_m)$ for the disjoint groups and by

repeatedly using required (t, n) threshold quantum secret sharing schemes. This idea is a modification of the ideas of chapter 5. The main methods of our paper are the multiple levels of shares obtained from the secret, random permutations and trap codes. The dealer maintains states according to the various combinations of the number of participant arriving from each group. For the quantum hierarchical scheme the groups of participants are first arranged as per the hierarchical order. The dealer first shares the secret in the highest level group. For the next lower level group, the dealer uses a random state to update the secret of the first(highest) group and shares it to the participants of the second lower level group. To update the secret the dealer uses random permutations and trap codes. This process continues till all the groups are exhausted. To reconstruct the secret, the lowest participants recover the updated secret of their generation. Using the random states, they recover the updated secrets of the generation one level above. This process goes on until they reach the highest level and the secret is recovered. For the other schemes, the dealer maintains different combinations of tuples and imposes different condition as per the access structure. Our schemes are general and can handle arbitrary number of participants. Hence we have constructed quantum versions of most known multipartite access structures existing in literature. Our scheme is simpler than the existing schemes, and the entangled state is not needed. The security of our scheme is based on the decoy particles, and the scheme can resist the popular quantum attacks. In the remarks we have shown ways to reduce the usage of quantum memory.

• *Open Problem :* The reduction of quantum memory mentioned in the remarks is not enough as Alice has to maintain the tuples to share the random states for all combinations of participants. Any way to reduce these tuples would lead to a different and more efficient method to realize these multipartite access structures.

10.5 Chapter 7

In this chapter we obtain robust/resilient secret sharing schemes by embedding function outputs of hard functions. This technique turns out to be helpful to reduce share sizes and in the cases of secure broadcasting and fine-grained analysis.

10.6 Ongoing Work

10.6.1 Perpetual Secret Sharing - A Quantum Version

We have seen in chapter 4 how to construct a "Perpetual Secret Sharing Scheme". This scheme helps overcome the drawbacks of the constructed schemes of chapter 3. We have also seen that quantum evolving schemes and the hierarchical schemes of chapters 5 and 6 use a huge amount of memory. Perpetual schemes are proven to be advantageous in reducing memory. Hence a quantum version of such a scheme can be helpful in reducing the memory requirements of the quantum versions of evolving, hierarchical and various multipartite schemes. In this context we refer the reader to quantum data structures [85], [71], [94].

10.7 Papers on which the thesis is based on

1. Samadder Chaudhury, S., *A Quantum Evolving Secret Sharing Scheme*, International Journal of Theoretical Physics, 59, pp. 3936–3950(2020), https://doi.org/10.1007/s10773-020-04644-5.

2. Samadder Chaudhury, S., Dutta, S. and Sakurai, K., *AC^0 constructions of secret sharing schemes: accommodating new parties*, 14th International Conference on Network and System Security Melbourne, Australia , 25-27 November '20, NSS 2020, LNCS 12570, pp. 292–308, https://doi.org/10.1007/978-3-030-65745-1-17.

3. Samadder Chaudhury, S., Dutta, S. and Sakurai, K., *Perpetual secret sharing from dynamic data structures*, The 2021 IEEE Conference on Dependable and Secure Computing, Aizuwakamatsu, Fukushima, Japan, 30th Jan - 2nd Feb, 2021 (DSC 2021), conference proceedings version to appear.

4. Samadder Chaudhury, S., Dutta, S. and Sakurai, K., *Embedding Hard Functions in Secret Sharing Schemes*, Submitted : Information Processing Letters, Manuscript Number : IPL-D-20-00074

5. Samadder Chaudhury, S. *A General Quantum Hierarchical Secret Sharing Scheme*, Submitted : International Journal of Quantum Information, Manuscript Number: IJQI-D-20-00143

6. Samadder Chaudhury, S. and Roy, B.K., *Generalized Matroid Ports and Non-Perfect Secret Sharing*, Revision under preparation, Designs, Codes and Cryptography, Manuscript Number : DESI-D-17-00422

Bibliography

[1] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51:595–605, 07 2004.

[2] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arXiv preprint arXiv:1704.04487*, 2017.

[3] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $\mathrm{ac}^0 \circ \mathrm{mod}_2$. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260, 2014.

[4] A. Ambainis. Polynomial degree vs. quantum query complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 230–239, 2003.

[5] Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

[6] Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(1):37–46, 2005.

[7] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[8] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography with constant input locality. *J. Cryptology*, 22(4):429–469, 2009.

[9] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[10] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. 2009.

[11] Chen-Ming Bai, Zhi-Hui Li, Meng-Meng Si, and Yong-Ming Li. Quantum secret sharing for a general quantum access structure. *The European Physical Journal D*, 71(10):1–8, 2017.

[12] Ming-Qiang Bai and Zhi-Wen Mo. Hierarchical quantum information splitting with eight-qubit cluster states. *Quantum information processing*, 12(2):1053–1064, 2013.

[13] Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:39, 2017.

[14] Li Bao-Kui, Yang Yu-Guang, and Wen Qiao-Yan. Threshold quantum secret sharing of secure direct communication. *Chinese Physics Letters*, 26(1):010302, 2009.

[15] Amos Beimel. Secret-sharing schemes: a survey. In *International Conference on Coding and Cryptology*, pages 11–46. Springer, 2011.

[16] Amos Beimel and Hussien Othman. Evolving ramp secret-sharing schemes. In *International Conference on Security and Cryptography for Networks*, pages 313–332. Springer, 2018.

[17] Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. In *Theory of Cryptography Conference*, pages 600–619. Springer, 2005.

[18] Claude Berge. Hypergraphs, north-holland mathematical library, 1989.

[19] A Bialostocki and P Dierker. Zero sum ramsey theorems. *Congressus Numerantium*, 70:119–130, 1990.

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

[21] George Robert Blakley and Catherine Meadows. Security of ramp schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 242–268. Springer, 1984.

[22] C. Blundo, A. Cresti, A. De Santis, and U. Vaccaro. Fully dynamic secret sharing schemes. In Douglas R. Stinson, editor, *Advances in Cryptology — CRYPTO' 93*, pages 110–125, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[23] Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 593–618, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[24] Andrej Bogdanov and Christopher Williamson. Approximate bounded indistinguishability. In *44th International Colloquium on Automata, Languages, and Pro-*

gramming, ICALP 2017, July 10-14, 2017, Warsaw, Poland, pages 53:1–53:11, 2017.

[25] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. pages 2105–2122, 10 2017.

[26] Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of Homomorphic Secret Sharing. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:21, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[27] Johannes Braun, Johannes Buchmann, Denise Demirel, Matthias Geihs, Mikio Fujiwara, Shiho Moriai, Masahide Sasaki, and Atsushi Waseda. Lincos: A storage system providing long-term integrity, authenticity, and confidentiality. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 461–468, New York, NY, USA, 2017. ACM.

[28] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *J. ACM*, 57(5):28:1–28:10, 2010.

[29] Ernest F Brickell and Daniel M Davenport. On the classification of ideal secret sharing schemes. *Journal of Cryptology*, 4(2):123–134, 1991.

[30] Anne Broadbent, Gus Gutoski, and Douglas Stebila. Quantum one-time programs. In *Annual Cryptology Conference*, pages 344–360. Springer, 2013.

[31] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low t-gate complexity. In *Annual Cryptology Conference*, pages 609–629. Springer, 2015.

[32] Anne Broadbent and Evelyn Wainewright. Efficient simulation for quantum message authentication. In *International Conference on Information Theoretic Security*, pages 72–91. Springer, 2016.

[33] Harry Buhrman and Ronald De Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288:2002, 2000.

[34] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 297–310, 2018.

[35] Mark Bun and Justin Thaler. A nearly optimal lower bound on the approximate degree of ac^0. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 1–12, 2017.

[36] Matteo Campanelli and Rosario Gennaro. Fine-grained secure computation. In *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II*, pages 66–97, 2018.

[37] Shion Samadder Chaudhury. A quantum evolving secret sharing scheme. *International Journal of Theoretical Physics*, pages 1–15, 2020.

[38] Shion Samadder Chaudhury, Sabyasachi Dutta, and Kouichi Sakurai. Hypercube and cascading-based algorithms for secret sharing schemes.

[39] Shion Samadder Chaudhury, Sabyasachi Dutta, and Kouichi Sakurai. Ac0 constructions of secret sharing schemes–accommodating new parties. In *International Conference on Network and System Security*, pages 292–308. Springer, 2020.

[40] Shion Samadder Chaudhury, Sabyasachi Dutta, and Kouichi Sakurai. Perpetual secret sharing from dynamic data structures. In *The 2021 IEEE Conference on Dependable and Secure Computing*, 2021.

[41] B Chazelle and L Guibas. Fractional cascading: I. a data structuring technique. *Algorithmica*, 1(1–4):133–162, 1986.

[42] B Chazelle and L Guibas. Fractional cascading: Ii. applications. *Algorithmica*, 1(1–4):163–191, 1986.

[43] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. Secure computation from random error correcting codes. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 291–310, 2007.

[44] Liqun Chen, Dieter Gollmann, and Chris J. Mitchell. Key escrow in mutually mistrusting domains. In *Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings*, pages 139–153, 1996.

[45] Weijiang Chen, Zekun Ye, and Lvzhou Li. Characterization of exact one-query quantum algorithms. *Physical Review A*, 101(2):022325, 2020.

[46] Xiu-Bo Chen, Gang Xu, Yuan Su, and Yi-Xian Yang. Robust variations of secret sharing through noisy quantum channel. *Quantum Information & Computation*, 14(7&8):589–607, 2014.

[47] Kuan Cheng, Yuval Ishai, and Xin Li. Near-optimal secret sharing and error correcting codes in ac0. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 424–458, Cham, 2017. Springer International Publishing.

[48] Kuan Cheng and Xin Li. Randomness extraction in AC0 and with small locality. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 37:1–37:20, 2018.

[49] Mahdi Cheraghchi. Nearly optimal robust secret sharing. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 2509–2513, 2016.

[50] Andrew M Childs. Secure assisted quantum computation. *arXiv preprint quant-ph/0111046*, 2001.

[51] Richard Cleve, Daniel Gottesman, and Hoi-Kwong Lo. How to share a quantum secret. *Physical Review Letters*, 83(3):648, 1999.

[52] Gil Cohen, Ivan Bjerre Damsgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae, 2013.

[53] David Conlon and Nina Kamcev. Intervals in the hales-jewett theorem. *arXiv preprint arXiv:1801.08919*, 2018.

[54] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 471–488, 2008.

[55] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 471–488, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[56] Harold Davenport. On the addition of residue classes. *Journal of the London Mathematical Society*, 1(1):30–32, 1935.

[57] Akshay Degwekar, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Fine-grained cryptography. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 533–562, 2016.

[58] Erik Demaine, Dion Harmon, John Iacono, and Mihai Pătraşcu. Dynamic optimality—almost. *SIAM Journal on Computing*, 37, 05 2007.

[59] Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. In *George Mason University, Tech. Report ISSE-TR-97-01, July 1997*. ftp://isse.gmu.edu/pub/techrep/97 01 jajodia.ps.gz., 1997.

[60] Yvo Desmedt and Kirill Morozov. Parity check based redistribution of secret shares. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 959–963, 2015.

[61] Matt DeVos, Luis Goddyn, and Bojan Mohar. A generalization of kneser's addition theorem. *Advances in Mathematics*, 220(5):1531–1548, 2009.

[62] Yu-Tao Du and Wan-Su Bao. Dynamic quantum secret sharing protocol based on two-particle transform of bell states. *Chinese Physics B*, 27(8):080304, 2018.

[63] Paul Erdos, Abraham Ginzburg, and Abraham Ziv. Theorem in the additive number theory. *Bull. Res. Council Israel F*, 10:41–43, 1961.

[64] Oriol Farràs, Torben Hansen, Tarik Kaced, and Carles Padró. Optimal non-perfect uniform secret sharing schemes. In *Annual Cryptology Conference*, pages 217–234. Springer, 2014.

[65] Oriol Farràs, Torben Brandt Hansen, Tarik Kaced, and Carles Padró. On the information ratio of non-perfect secret sharing schemes. *Algorithmica*, 79(4):987–1013, 2017.

[66] Oriol Farras, Jaume Martí-Farré, and Carles Padró. Ideal multipartite secret sharing schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 448–465. Springer, 2007.

[67] Oriol Farras, Sebastià Martín Molleví, and Carles Padró. A note on non-perfect secret sharing. *IACR Cryptol. ePrint Arch.*, 2016:348, 2016.

[68] Oriol Farras and Carles Padró. Ideal hierarchical secret sharing schemes. *IEEE transactions on information theory*, 58(5):3273–3286, 2012.

[69] Oriol Farras and Carles Padró. Extending brickell–davenport theorem to non-perfect secret sharing schemes. *Designs, Codes and Cryptography*, 74(2):495–510, 2015.

[70] Oriol Farras, Carles Padró, Chaoping Xing, and An Yang. Natural generalizations of threshold secret sharing. *IEEE transactions on information theory*, 60(3):1652–1664, 2014.

[71] Maximilian Fillinger. Data structures in classical and quantum computing, 2013.

[72] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal resilience proactive public-key cryptosystems. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 384–393, 1997.

[73] Saioru Fujishige. Entropy functions and polymatroids-combinatorial structures in information-theory. *ELECTRONICS & COMMUNICATIONS IN JAPAN*, 61(4):14–18, 1978.

[74] Zoltán Füredi and Daniel J Kleitman. On zero-trees. *Journal of graph theory*, 16(2):107–120, 1992.

[75] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[76] Mike Furst, Jon Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.

[77] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum. Verifying and decoding in constant depth. In *Proceedings of the 39th*

Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007, pages 440–449, 2007.

[78] Daniel Gottesman. Theory of quantum secret sharing. *Physical Review A*, 61(4):042311, 2000.

[79] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. In *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, volume 68, pages 13–58, 2010.

[80] Markus Grassl and Martin Roetteler. Quantum error correction and fault tolerant quantum computing. *Computational Complexity: Theory, Techniques, and Applications*, pages 2478–2495, 11 2013.

[81] David J Grynkiewicz. *Structural additive theory*, volume 30. Springer Science & Business Media, 2013.

[82] Venkatesan Guruswami and Adam D. Smith. Optimal rate code constructions for computationally simple channels. *J. ACM*, 63(4):35:1–35:37, 2016.

[83] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.

[84] Torben Hagerup. Fast parallel generation of random permutations. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, page 405–416, Berlin, Heidelberg, 1991. Springer-Verlag.

[85] Vladimir Hahanov, Irina Hahanova, Olesya Guz, and Murad Ali Abbas. Quantum models for data structures and computing. In *Proceedings of International Conference on Modern Problem of Radio Engineering, Telecommunications and Computer Science*, pages 291–291. IEEE, 2012.

[86] Alfred W Hales and Robert I Jewett. Regularity and positional games. In *Classic Papers in Combinatorics*, pages 320–327. Springer, 2009.

[87] Yahya Ould Hamidoune. Subsequence sums. *Combinatorics, Probability & Computing*, 12(4):413, 2003.

[88] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986.

[89] Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014.

[90] Javier Herranz and Germán Sáez. New results on multipartite access structures. *IEE Proceedings-Information Security*, 153(4):153–162, 2006.

[91] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology — CRYPT0' 95*, pages 339–352, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[92] Jung-Lun Hsu, Song-Kong Chong, Tzonelih Hwang, and Chia-Wei Tsai. Dynamic quantum secret sharing. *Quantum Information Processing*, 12(1):331–344, 2013.

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

[94] Stacey Jeffery, Robin Kothari, and Frédéric Magniez. Nested quantum walks with quantum data structures. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1474–1485. SIAM, 2013.

[95] Eyal Kaplan, Moni Naor, and Omer Reingold. Derandomized constructions of k-wise (almost) independent permutations. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 354–365, 2005.

[96] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993.

[97] Emilia Käsper, Ventzislav Nikov, and Svetla Nikova. Strongly multiplicative hierarchical threshold secret sharing. In *International Conference on Information Theoretic Security*, pages 148–168. Springer, 2007.

[98] Adrian Keet, Ben Fortescue, Damian Markham, and Barry C Sanders. Quantum secret sharing with qudit graph states. *Physical Review A*, 82(6):062315, 2010.

[99] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. In *Theory of Cryptography Conference*, pages 485–514. Springer, 2016.

[100] Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: dynamic thresholds and robustness. In *Theory of Cryptography Conference*, pages 379–393. Springer, 2017.

[101] Hugo Krawczyk. Secret sharing made short. *Advances in Cryptology - CRYPTO 93*, pages 136–146, 1994.

[102] Jun Kurihara, Shinsaku Kiyomoto, Kazuhide Fukushima, and Toshiaki Tanaka. On a fast $(k,\ n)$-threshold secret sharing scheme. *IEICE Transactions*, 91-A(9):2365–2378, 2008.

[103] Kaoru Kurosawa, Koji Okada, Keiichi Sakano, Wakaha Ogata, and Shigeo Tsujii. Nonperfect secret sharing schemes and matroids. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 126–141. Springer, 1993.

[104] Russell W. F. Lai, Giulio Malavolta, and Dominique Schröder. Homomorphic secret sharing for low degree polynomials. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, pages 279–309, 2018.

[105] S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using kolmogorov arguments. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 294–304, 2004.

[106] Sang Min Lee, Seung-Woo Lee, Hyunseok Jeong, and Hee Su Park. Quantum teleportation of shared quantum secret. *Physical Review Letters*, 124(6):060501, 2020.

[107] Alfred Lehman. A solution of the shannon switching game. *Journal of the Society for Industrial and Applied Mathematics*, 12(4):687–725, 1964.

[108] Ci-Hong Liao, Chun-Wei Yang, and Tzonelish Hwang. Dynamic quantum secret sharing protocol based on ghz state. *Quantum Information Processing*, 13(8):1907–1916, 2014.

[109] F Liu, C Wu, and X Lin. Step construction of visual cryptography schemes. *IEEE Transactions on Information Forensics and Security*, 5(1):27–38, 2010.

[110] Shachar Lovett and Emanuele Viola. Bounded-depth circuits cannot sample good codes. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 243–251, 2011.

[111] He Lu, Zhen Zhang, Luo-Kan Chen, Zheng-Da Li, Chang Liu, Li Li, Nai-Le Liu, Xiongfeng Ma, Yu-Ao Chen, and Jian-Wei Pan. Secret sharing of a quantum state. *Physical review letters*, 117(3):030501, 2016.

[112] Frédéric Magniez, Ashwin Nayak, Miklos Santha, and David Xiao. Improved bounds for the randomized decision tree complexity of recursive majority, 2010.

[113] Arpita Maitra and Goutam Paul. A resilient quantum secret sharing scheme. *International Journal of Theoretical Physics*, 54(2):398–408, 2015.

[114] Jaume Martí-Farré and Carles Padró. On secret sharing schemes, matroids and polymatroids. In *Theory of Cryptography Conference*, pages 273–290. Springer, 2007.

[115] Yossi Matias and Uzi Vishkin. Converting high probability into nearly-constant time—with applications to parallel hashing. 1991.

[116] M. Minsky and S. Papert. *Perceptrons*. MIT Press, 1969.

[117] Sandeep Mishra, Chitra Shukla, Anirban Pathak, R Srikanth, and Anu Venugopalan. An integrated hierarchical dynamic quantum secret sharing protocol. *International Journal of Theoretical Physics*, 54(9):3143–3154, 2015.

[118] Paz Morillo, Carles Padró, Germán Sáez, and Jorge Luis Villar. Weighted threshold secret sharing schemes. *Information processing letters*, 70(5):211–216, 1999.

[119] Michele Mosca, Alain Tapp, and Ronald de Wolf. Private quantum channels and the cost of randomizing quantum information. *arXiv preprint quant-ph/0003101*, 2000.

[120] Moni Naor and Adi Shamir. Visual cryptography. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, pages 1–12, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[121] Melvyn B Nathanson. *Additive number theory: Inverse problems and the geometry of sumsets*, volume 165. Springer Science & Business Media, 1996.

[122] S-L Ng. Ideal secret sharing schemes with multipartite access structures. *IEE Proceedings-Communications*, 153(2):165–168, 2006.

[123] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

[124] Mehrdad Nojoumian and Douglas R. Stinson. On dealer-free dynamic threshold schemes. *Adv. in Math. of Comm.*, 7(1):39–56, 2013.

[125] Mehrdad Nojoumian and Douglas R. Stinson. Sequential secret sharing as a new hierarchical access structure. *J. Internet Serv. Inf. Secur.*, 5(2):24–32, 2015.

[126] Ryan O'Donnell. *Analysis of Boolean Functions*. 2012.

[127] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

[128] Huawang Qin and Yuewei Dai. Verifiable (t, n) threshold quantum secret sharing using d-dimensional bell state. *Information Processing Letters*, 116(5):351–355, 2016.

[129] Huawang Qin and Yuewei Dai. Dynamic quantum secret sharing by using d-dimensional ghz state. *Quantum information processing*, 16(3):64, 2017.

[130] Huawang Qin, Wallace KS Tang, and Raylin Tso. Hierarchical quantum secret sharing based on special high-dimensional entangled state. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(3):1–6, 2020.

[131] Huawang Qin, Xiaohua Zhu, and Yuewei Dai. (t, n) threshold quantum secret sharing using the phase shift operation. *Quantum Information Processing*, 14(8):2997–3004, 2015.

[132] Michael Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36:335–348, 1989.

[133] Alexander Razborov. Lower bounds for the size of circuits of bounded depth with basis XOR,OR. *Math. notes of the Academy of Science of the USSR*, 41(4):333–338, 1987.

[134] Ben W Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 544–551. IEEE, 2009.

[135] Alexander Schrijver and Paul D Seymour. Spanning trees of different weights. In *Polyhedral combinatorics*, page 281, 1990.

[136] Alexander Schrijver and Paul D Seymour. A simpler proof and a generalization of the zero-trees theorem. *Journal of Combinatorial Theory, Series A*, 58(2):301–305, 1991.

[137] Paul D Seymour. A note on the production of matroid minors. *Journal of Combinatorial Theory, Series B*, 22(3):289–295, 1977.

[138] PD Seymour. A forbidden minor characterization of matroid ports. *The Quarterly Journal of Mathematics*, 27(4):407–413, 1976.

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

[140] Kunal Sharma, Eyuri Wakakuwa, and Mark M Wilde. Conditional quantum one-time pad. *Physical Review Letters*, 124(5):050503, 2020.

[141] Chitra Shukla and Anirban Pathak. Hierarchical quantum communication. *Physics Letters A*, 377(19-20):1337–1344, 2013.

[142] Gustavus J Simmons. How to (really) share a secret. In *Conference on the Theory and Application of Cryptography*, pages 390–448. Springer, 1988.

[143] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *In Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC '87*, pages 77–82, 1987.

[144] Marc Snir. Lower bounds for probabilistic linear decision trees. *Theoretical Computer Science*, 38:69–82, 1985.

[145] Douglas R. Stinson. An explication of secret sharing schemes. *Designs, Codes and Cryptography*, 2(4):357–390, 1992.

[146] Douglas R. Stinson and Ruizhong Wei. Unconditionally secure proactive secret sharing scheme with combinatorial structures. In *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings*, pages 200–214, 1999.

[147] Tamir Tassa. Hierarchical threshold secret sharing. *Journal of cryptology*, 20(2):237–264, 2007.

[148] Tamir Tassa and Nira Dyn. Multipartite secret sharing by bivariate interpolation. *Journal of Cryptology*, 22(2):227–258, 2009.

[149] Emanuele Viola. On approximate majority and probabilistic time. *Computational Complexity*, 18(3):337–375, 2009.

[150] Emanuele Viola. The complexity of distributions. *SIAM Journal on Computing*, 41(1):191–218, 2012.

[151] Emanuele Viola. The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012.

[152] Heribert Vollmer. *Introduction to Circuit Complexity a Uniform Approach*. 1999.

[153] Xin-Wen Wang, Li-Xin Xia, Zhi-Yong Wang, and Deng-Yu Zhang. Hierarchical quantum-information splitting. *Optics communications*, 283(6):1196–1199, 2010.

[154] Xin-Wen Wang, Deng-Yu Zhang, Shi-Qing Tang, and Li-Jun Xie. Multiparty hierarchical quantum-information splitting. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 44(3):035505, 2011.

[155] Xin-Wen Wang, Deng-Yu Zhang, Shi-Qing Tang, and Li-Jun Xie. Multiparty hierarchical quantum-information splitting. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 44(3):035505, 2011.

[156] Xin-Wen Wang, Deng-Yu Zhang, Shi-Qing Tang, Xiao-Gui Zhan, and Kai-Ming You. Hierarchical quantum information splitting with six-photon cluster states. *International Journal of Theoretical Physics*, 49(11):2691–2697, 2010.

[157] YuJue Wang, QianHong Wu, Duncan S Wong, Bo Qin, Yi Mu, and JianWei Liu. Further ideal multipartite access structures from integer polymatroids. *Science China Information Sciences*, 58(7):1–13, 2015.

[158] DJA Welsh. Matroid theory academic press london. *New-York, San Francisco*, 1976.

[159] Gang Xu, Cong Wang, and Yi-Xian Yang. Hierarchical quantum information splitting of an arbitrary two-qubit state via the cluster state. *Quantum information processing*, 13(1):43–57, 2014.

[160] Xin-Wei Zha, Ning Miao, and Hui-Fang Wang. Hierarchical quantum information splitting of an arbitrary two-qubit using a single quantum resource. *International Journal of Theoretical Physics*, 58(8):2428–2434, 2019.

[161] Yanshuo Zhang, Zhuojun Liu, and Guifang Huang. Sure interpolation and its application to hierarchical threshold secret sharing scheme. In *2008 International Symposium on Computer Science and Computational Technology*, volume 1, pages 447–450. IEEE, 2008.

[162] Zhan-jun Zhang, Yong Li, and Zhong-xiao Man. Multiparty quantum secret sharing. *Physical Review A*, 71(4):044301, 2005.

Appendix A

List of Symbols

Symbol	Definition/Description
$\mathbb{N}$	Set consisting of all natural numbers.
$\mathbb{R}$	Set consisting of all real numbers.
$\mathbb{R}^n$	Euclidean vector space of dimension n.
$\mathbb{R}^{+n}$	Vector space of dimension n, each element being a positive number.
$((p_{ij}))_{m\times n}$	A matrix with m rows and n columns. First row has elements $p_{11}, p_{12}, \cdots, p_{1n}$, second row has elements $p_{21}, p_{22}, \cdots, p_{2n}$ and similarly others up to m^{th} row which has elements $p_{m1}, p_{m2}, \cdots, p_{mn}$.
$A \backslash B$	Set consisting of those elements of A that are not in B.
$A \subseteq B$	A is a subset of B.
$\|x\|$	Absolute value of the number x.
$sign(x)$	Sign of the number x.
$K : A \to B$	Function K mapping the set A to B.

Symbol	Definition/Description
(a, b)	The interval $\{x \in \mathbb{R} : a \leq x \leq b\}$.
$[a, b]$	The interval $\{x \in \mathbb{R} : a < x < b\}$.
$(a, b]$	The interval $\{x \in \mathbb{R} : a \leq x < b\}$.
$[a, b)$	The interval $\{x \in \mathbb{R} : a < x \leq b\}$.
$\Longrightarrow$	Implies
$\approx$	Approximately equal to
$\xrightarrow{P}$	Converges in Probability.
$\binom{n}{k}$	Binomial coefficient $n!/[k!(n-k)!]$.
$\lim_{x \to a} f(x)$	Limit of function f at a.
sup	Supremum.
$\min(x \in A : B)$	Minimum of all values in set A such that event B holds.
$f(.)$	Function f.
$P(A)$	Probability measure of the set A.
$P(A \mid B)$	Conditional Probability measure of A given B
$[m]$	set of natural numbers $1, 2, \ldots, m$