AC^0 Constructions for Evolving Secret Sharing Schemes and Redistribution of Secret Shares

Shion Samadder Chaudhury[1], Sabyasachi Dutta[2], and Kouichi Sakurai[3]

[1] Applied Statistics Unit,
Indian Statistical Institute, Kolkata, India
`chaudhury.shion@gmail.com`
[2] Department of Computer Science,
University of Calgary, Calgary, Canada
`saby.math@gmail.com`
[3] Faculty of Information Science and Electrical Engineering,
Kyushu University, Japan
`sakurai@inf.kyushu-u.ac.jp`

Abstract. Classical secret sharing schemes are built on the assumptions that the number of participants and the access structure remain fixed over time. Evolving secret sharing addresses the question of accommodating new participants with changeable access structures. One goal of this article is to initiate the study of evolving secret sharing sharing such that both share generation and reconstruction algorithms can be implemented by AC^0 circuits. We give a concrete construction with some minor storage assumption. Furthermore, allowing certain trade-offs we consider the novel problem of robust redistribution of secret shares (in AC^0) in the spirit of dynamic access structure by suitably modifying a construction of Cheng-Ishai-Li (TCC 2017). A naive solution to the problem is to increase the alphabet size. We avoid this by modifying shares of some of the old participants. This modification is also necessary to make newly added participant(s) non-redundant to the secret sharing scheme.

Keywords: Secret sharing, access structure, evolving access structure, dynamic secret sharing, AC^0 complexity class

1 Introduction

Secret sharing is a method to distribute a secret piece of information among n many participants so that any predefined "qualified" sets of participants can recover the secret information, whereas every predefined "forbidden" sets of participants do not get any information about the secret. The *monotone* collection of qualified sets of participants is called an *access structure*. Secret sharing schemes have found applications in cryptography as well as in secure distributed computing. So studying secret sharing scheme is, in itself very important. All the classical secret sharing schemes assume that the number of participants is fixed, as well as the access structure is well-defined beforehand. An access structure is called an *evolving* access structure if any one (or both) of the above two presumptions fail to hold. So needless to say that the number of participants can be potentially infinite and existing classical methodology fails to provide a secret sharing scheme when the access structure is evolving. Some recent works (*see* Section 1.2) have put forward secret sharing schemes for evolving access structures. In the same time frame, almost parallely, 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 constant-depth, polynomial size circuits which consists of the AND, OR and NOT gates with the AND and OR gates having unbounded fan-in. In other words, secret sharing is possible with both share-generation algorithm and reconstruction algorithms are in the complexity class AC^0.

1.1 Motivation

The motivation for this paper comes from the above two sources. On one hand we see that it is possible to construct secret sharing schemes for evolving access structures and on the other, there are robust schemes implementable in the class AC^0 for certain predefined access structures. We consider the problem of whether it is possible to implement secret sharing schemes in AC^0 for evolving access structures while making it robust against an adversary - even a possibility would motivate further research along this direction. On the other hand, we consider a practical situation when a dealer distributes a secret among several participants and then goes offline. Anticipating threats from a malicious adversary, the participants may want to redistribute their shares among some new participants without actually recovering the secret. Hence it is important to implement a secret sharing scheme in AC^0(a very low complexity class) where the access structure changes over time. We make a quick remark that the existing works in secret sharing (with some extra functionality) in the literature make use of basic primitives e.g. linear algebraic computation [28, 6], parity [18, 22] etc. which cannot be implemented in AC^0.

1.2 Related work

Classical Secret Sharing Secret sharing schemes were proposed independently by Shamir [28] and Blakley [6] 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 resulting access structure is called a (k, n)-threshold access structure where k acts as a threshold value for being qualified. Both schemes were fairly efficient in terms of the size of the shares and computational complexity. Ito et al. [18] showed that it is possible to construct secret sharing schemes for any monotone (general) access structures but with exponential share sizes. Later, Karchmer et al. [19] 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.

Evolving Secret Sharing The classical secret sharing schemes assume that the number of participants and the access structure is known in advance. Komargodski et al. [20] introduced evolving secret sharing schemes where the *dealer* does not know in advance, the number of participants that will participate and no upper bound on their number. Thus, number of participants could be potentially infinite and the access structure may change with time. Komargodski et al. [20] considered the scenario when participants come one by one and receives their share from the dealer; the dealer however cannot update the shares that he has already distributed. The authors showed that for every evolving access structure there exists a secret sharing scheme where the share size of the t^{th} participant is 2^{t-1}. They also constructed (k, ∞)-threshold evolving secret sharing scheme for constant k in which the share size of the t^{th} participant is $(k-1)logt+\mathcal{O}(loglogt)$. Furthermore, they have provided an evolving 2-threshold scheme which is nearly optimal in the share size of the t^{th} participant viz. $logt + \mathcal{O}(loglogt)$. Later, Komargodski and Paskin-Cherniavsky [21] forwarded the idea of evolving k-threshold schemes to evolving dynamic threshold schemes and provided a secret sharing scheme in which the share size of the t^{th} participant is $\mathcal{O}(t^4 logt)$ bits. Furthermore, they showed how to transform evolving threshold secret sharing schemes to *robust* schemes with the help of *algebraic manipulation detection* (AMD) codes. 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. A very recent work by Beimel and Othman [5] considers the problem of ramp secret sharing for evolving threshold schemes and drastically reduced the share size to $\mathcal{O}(1)$. Beimel and Othman [5], construct evolving (a, b) ramp scheme 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.

Secret Sharing in $\boldsymbol{AC^0}$ 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 [28] 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 [25] require only computation of OR function which can be implemented by AC^0 circuit. Recent work by Bogdanov et al. [7] considers the question of 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. [7] was followed up by a work of Cheng et al. [10] 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 $\frac{1}{poly(n)}$.

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. [9], Frankel et al. [16] and Desmedt-Jajodia [14].

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.

Nojoumian-Stinson [26] proposed unconditionally secure share re-distribution schemes, in absence of a dealer, based on a previously existing VSS protocol of Stinson-Wei [29]. 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 [15] 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 [27] 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.

Secure Computation against moderately complex adversaries An important requirement of cryptography is to protect not only information but also the computations that are performed on data. The traditional cryptographic approach has been based on computational tasks which are easy for the 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.[13] 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]. In the paper by Ball et al.[4], they present computational problems which are "moderately hard" on average. Continuing in this line Campanelli and Gennaro [8] 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 [10].

1.3 Our Contribution

In this paper we make the following contributions:

1. We first consider the possibility of existence of an evolving secret sharing scheme which can be implemented in AC^0 which is the lowest complexity class for which a secret can be reconstructed or a message be decoded with negligible error probability. We give an explicit construction of an AC^0 secret sharing scheme such that any two participants are authorized to recover the secret and the number of participants can be potentially infinite.
2. Next, we give the construction of a scheme which extends the robust AC^0 secret sharing scheme in [10] to accommodate a bounded number of new participants (in the sense of dynamic secret sharing schemes where one can add new participants over time). Our novel idea is : during accommodating new participants we have to modify the shares of some of the old participants. As we see that this is necessary to keep the scheme AC^0 computable and not to make the new participants redundant to the scheme. An advantage of modifying old shares is that we do not have to increase the alphabet size to accommodate new participants. On the downside, we can only accommodate a bounded number of new participants and when the capacity to accommodate the number of new participants is exhausted, the whole system has to be refreshed again.
3. We consider the question of re-distribution of secret shares in the absence of dealer. For example suppose after the dealer distributes the secret, he creates a hierarchy among the participants who can now modify old shares to accommodate new participants. This has a similarity with (hierarchical) sequential secret sharing and is applicable in practical scenarios.

In parts 2 and 3 we note that the schemes are robust under non-adaptive adversaries. While various secret sharing schemes have been proposed where the access structures change over time, the authors are not aware of attempts to make such schemes both robust and AC^0 computable. This also opens up a future line of work where one can consider low complexity evolving(dynamic) secret sharing schemes robust under NC^1 adversaries, adaptive adversaries, dishonest parties, reducing reconstruction error and so on

Tradeoffs The existing constructions of evolving or dynamic or sequential secret sharing schemes are not AC^0 computable since they use linear algebraic methods which are not in AC^0. The construction technique that we follow does not exactly implement these schemes in AC^0. Rather we take a middle ground with certain tradeoffs which helps us to incorporate key features of the abovementioned secret sharing schemes in our construction. For example in the case of evolving secret sharing, the scheme is able to handle possibly infinitely many new participants. To make our construction of a robust scheme where the number of participants increase over time AC^0 computable, we can only handle a bounded collection of new participants. Also for the construction of our evolving scheme in AC^0, we make a storage assumption. In this paper we make no attempt to reduce the share size of the secret sharing schemes or to construct schemes robust under adaptive adversaries.

2 Preliminaries

We discuss some basic definitions and results that will be needed throughout the paper. We mainly adopt the notations and definitions of [7, 20].

For a positive integer n the set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. Let $\mathcal{P}_n = [n]$ be a set of n participants. Let $2^{\mathcal{P}_n}$ denote the power set of $\mathcal{P}_n$. A collection $\mathcal{A} \subset 2^{\mathcal{P}_n}$ is said to be *monotone* if $A \in \mathcal{A}$ and $A \subset B$ imply $B \in \mathcal{A}$.

Definition 1. *(Access structure) $\mathcal{A} \subset 2^{\mathcal{P}_n}$ is called a monotone access structure if the collection $\mathcal{A}$ is monotone. Any subset A of $\mathcal{P}_n$ which are in $\mathcal{A}$ are called qualified sets and $F \notin \mathcal{A}$ are called forbidden.*

Definition 2. *(Threshold Access structure) Let $n \in \mathbb{N}$ and $0 < k \leq n$. A (k, n)-threshold access structure $\mathcal{A}$ on a participant set $[n]$ is defined by $\mathcal{A} = \{X \subset [n] : |X| \geq k\}$.*

We now define restriction of an access structure to its first $m < n$ participants which in essence describes the qualified sets formed by the participants in $[m]$ in $\mathcal{A}$.

Definition 3. *(Restriction of Access structure) Let $\mathcal{A}_n$ be an access structure on a set of n participants $\mathcal{P}_n = [n]$ and let $1 \leq m \leq n-1$. The restriction of the given access structure to the first m participants, denoted by $\mathcal{A}_n|_m$, is defined to be the collection $\mathcal{A}_n|_m = \{X \in \mathcal{A}_n : \{m+1, m+2, \ldots, n\} \cap X = \emptyset\}$.*
If it is clear from the context that $\mathcal{A}_n$ is an access structure on the participant set $[n]$ then we drop the suffix n and simply write $\mathcal{A}$.

Definition 4. *(Evolving Access structure) An infinite sequence of access structures $\{\mathcal{A}_i\}_{i \in \mathbb{N}}$ is called an evolving access structure if:*

1. *for every $i \in \mathbb{N}$, $\mathcal{A}_i$ is an access structure over $[i]$.*
2. *for every $i \geq 2$, $\mathcal{A}_i|_{i-1} = \mathcal{A}_{i-1}$.*

We now give the definition of evolving threshold access structure. We define it in the sense of dynamic thresholds [21] where the threshold values change over time. It is to be noted that in [20] an evolving threshold access structure is defined so that the threshold value is fixed but the number of participants increases.

Definition 5. *(Evolving Threshold Access structure) A dynamic threshold access structure is parametrized by an infinite sequence of thresholds $k_1 \leq k_2 \leq \dots$ such that for any $t \in \mathbb{N}$, the access structure is defined by $\mathcal{A}_t = \{X : |X| \geq k_t\}$.*

Remark 1. We note that $k_1 = k_2 = \dots = k$ can be a fixed value. This gives us an evolving access structure where the threshold value is fixed but the number of participants increases in an unbounded manner. We denote such an evolving threshold access structure by (k, ∞)- threshold access structure.

2.1 Secret Sharing Scheme

In a secret sharing scheme there is a dealer who has a secret s, a set of participants $[n]$ and an access structure $\mathcal{A}$. The dealer shares the secret among the participants in such a way that any qualified set of participants can recover the secret but any forbidden set of participants has no information about the secret.

Definition 6. *(Secret Sharing Scheme) A secret sharing scheme $\mathcal{S}$ for an access structure $\mathcal{A}$ consists of a pair of algorithms $(Share, Rec)$. Share is a probabilistic algorithm that gets as input a secret s (from a domain of secrets S) and a number n, and generates n shares $\Pi_1^{(s)}, \Pi_2^{(s)}, \dots, \Pi_n^{(s)}$. Rec is a deterministic algorithm that gets as input the shares of a subset B of participants and outputs a string. The requirements for defining a secret sharing scheme are as follow:*

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

The *share size* of a secret sharing scheme $\mathcal{S}$ is the maximum number of bits each participant has to hold in the worst case over all participants and all secrets.

Definition 7. *(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 8. *(Evolving Secret Sharing Scheme) 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.*

2.2 Secret sharing scheme in AC^0

Let Σ denote the 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 wee 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 9. *(Secret Sharing for 1-bit secret) An (n, k, r) bit secret sharing scheme with alphabet Σ, reconstruction function $f : \Sigma^r \to \{0, 1\}$ 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 $Pr[f(\mu|_S) = 1] - Pr[f(\nu|_S) = 1] \geq \alpha$.*

2.3 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 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 [2, 3, 1, 4, 23] have been done focusing on possibilities of obtaining cryptographic primitives in low complexity classes e.g. AC^0 or NC^1. We 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,

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

We say that X is $SD(X;Y)$-close to Y.

Minsky-Papert CNF function

The sharing function, Share, used in our construction is based on the CNF function given by Minsky-Papert [24]. This scheme can share one bit among n participants, with binary alphabet, privacy threshold $\Omega(n^{1/3})$ and perfect reconstruction.

Robust Secret Sharing Scheme

The $Share_C$ function in the construction is the same as the sharing function for secret sharing schemes based on error correcting codes. the construction first amends the secret with a tag using an AMD code (such as the one in [12]). Then, it uses Shamir's scheme to encode the result into mn shares, for a carefully chosen integer parameter $m > 1$. Finally, the resulting shares are bundled into n groups of size m each which are distributed among the n participants. In other words, we use a variant of Shamir's scheme based on folded Reed-Solomon codes (instead of plain Reed-Solomon codes) combined with an AMD pre-code. This is used to provide robustness in the sense of error-detection.

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 [17]. A set of n random variables, $X_1, ..., X_n$, 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 $g(U) = \{Y_1, ..., Y_n\}$ 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 Main Results

In this Section we first show the existence of secret sharing scheme implementable in AC^0 which realizes a $(2, \infty)$-evolving threshold access structure. Secondly, we give a dynamic robust secret sharing scheme in AC^0 which can be achieved in the absence of the dealer.

3.1 AC^0 construction of $(2, \infty)$-secret sharing scheme

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 $(Share_+, Rec_+)$ be a 2-out-of-2 threshold secret sharing scheme which can be obtained using the techniques of [7]. Applying this $(Share_+, Rec_+)$ algorithm, multiple times we show how the dealer prepares the shares for the participants.

Algorithm :

1. The dealer first applies $Share_+$ algorithm on s and outputs (s_1^1, s_1^2).
2. The dealer gives s_1^1 to the participant 1.
3. When participant 2 arrives, the dealer again runs $Share_+(s) \longrightarrow (s_2^2, s_2^3)$. Share of 2 is (s_1^2, s_2^2) and dealer stores s_2^3.
4. When participant 3 arrives, the dealer again runs $Share_+(s) \longrightarrow (s_3^3, s_3^4)$. Share of 3 is (s_1^2, s_2^3, s_3^3) and stores the value s_3^4.
5. 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 $Share_+(s)$ 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 $(s_t^t, s_t^{t+1}) \longleftarrow Share_+(s)$ run for the t-th time.

   Figure 1 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) $Share_+$ algorithm run independently every time with the fixed secret bit s as input.

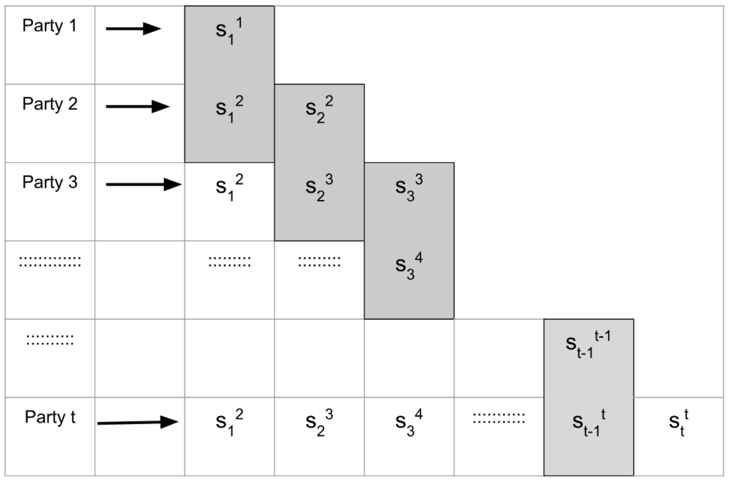

Fig. 1: Distribution of shares using a basic 2-out-of-2 AC^0 secret sharing scheme.

The reconstruction algorithm is simple. When two participants come together they produce only the corresponding shares that connects them.

Theorem 1. *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 circuits.

To prove that any two participants can recover the secret let us suppose that participants i, 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, $(s_j^j, s_j^{j+1}) \longleftarrow Share_+(s)$ 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 2. 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 [10]. It is not very hard to see that the above construction gives a secret sharing sharing scheme with the same privacy error as the basic one.

Example 1. (Yet another example)
Let us consider a star-graph based access structure where the internal node is fixed over time but the number of leaves changes/increases over time. A minimal

qualified set is defined by two nodes which has an edge between them. More precisely, $\{fixed\ internal\ node,\ any\ leaf\}$ constitutes a minimal qualified set. Let $(Share_+, Rec_+)$ be a 2-out-of-2 threshold secret sharing scheme in which from any two shares the secret can be reconstructed and any one share does not reveal information about the secret. The dealer runs $Share_+(s)$ (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.

3.2 A dynamic threshold secret sharing scheme in AC^0

Our goal is to construct dealer-free redistribution of shares which is AC^0 computable. In particular, we apply the techniques of Cheng et al. [10] and modify it to accommodate participants who come one by one. Our proofs are in similar line with those of [10], so we give the overview of our proofs. First we discuss the method of Cheng et al. to construct robust secret sharing schemes in AC^0.

Overview of technique In this section we first give an overview of the technique to construct robust secret sharing schemes in AC^0. The following techinque is used in [10]. Suppose there is a short random seed R. This random seed is shared using the one-in-a-box function to get n-participants with privacy threshold k_0. One then uses R and the k-wise independent generator to generate an n-bit string Y. To share a secret X, only $Y \oplus X$ is computed. To reconstruct the secret, the n participants are used to reconstruct R, compute Y and then to compute X. This whole procedure can be computed in AC^0. To boost the privacy threshold and make the scheme robust the following steps are taken:

1. *The participants are divided into blocks of size $O(log^2 n)$.*
2. *For each block a secret sharing scheme based on asymptotically good error-correcting codes is applied to obtain $O(log^2 n)$ shares.*
3. *These shares are further divided into $O(logn)$ smaller blocks of size $O(logn)$ 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.*

Suppose the adversary sees only 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, the fact that there are two levels of blocks and by adjusting the parameters, 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. To extend to robust secret sharing schemes, the robust schemes of Cheraghchi [11] are used in the first and second blocks. One issue is that the adversary can modify some of the indices. This information is necessary during reconstruction as a random permutation is applied. To avoid this situation the index is stored multiple times in the second block. By adjusting the parameters it can be ensured that at least $2/3rd$ of the inputs are the same and we can use *approximate majority* during reconstruction which can be computed in AC^0.

Accommodating new participants: Suppose at a particular instance we have a robust secret sharing scheme as before with n participants which is AC^0 computable. In the next instance there are two cases- (a) One new participant arrives and (b) a set of new participants arrives.

- In the first case where only one new participant arrives, the following steps are taken:
  1. *Add it in any of the larger blocks. Adding a new participant in the larger block keeps the size of the block $O(log^2 n)$.*
  2. *Store the additional informations like the generation of the new participant and to which block it is added multiples times and proceed as before, i.e., step 3 of previous robust scheme.*

  The procedure to give this new participant a share is discussed later. This whole procedure can be computed in AC^0. Since the information of the new participant is stored multiple times, approximate majority can be used during the reconstruction part which is AC^0 computable.
- In the second case when a collection of new participant arrives, an arbitrary number of new participants cannot be accommodated since we need the whole procedure to be AC^0 computable. The critical step which keeps the procedure AC^0 computable is the division of the participants and the shares into small blocks of size $O(\log^2 n)$ and $O(\log n)$ respectively. So when a collection of new participants arrive, keep adding them to a block until the size remains $O(\log^2 n)$ and then proceed to the next block. This gives an upper bound on the number of new participants that one can accommodate to keep the operations AC^0 computable. Also by extending the alphabet size, informations like the generation of a participant, to which block a participant is stored multiple times to keep the scheme robust. Once the participants are added to the respective blocks, the respective shares are computed (the method is discussed below) to get an AC^0 computable evolving access structure which can accommodate a bounded collection of new participants. When any more new participant cannot be accommodated, we refresh the whole system. That is, all the available participants are taken , old and new, as a set of participants and the whole system is constructed afresh. The new number of participants is n added to the number of new participants that arrived.

Generating a share of a new participant: To construct a share for the new participant, our idea as follows :

1. Select a participant from the old set of participants, say A.
2. Take the first half of the share of A and copy it as the first half of the share of the new participant.
3. Choose another old participant from the remaining participants, say B, copy the first half of its share and assign it as the first half of the share of A and the second half of the share of the new participant.
4. Hence the new share of A is the first half of the share of B concatenated with the second half of the original share of A.

5. The share of the new participant is the first half of the original share of A concatenated to the first half of the share of B.
6. Store the generation, the participant whose share is being copied, and to which block the new participant(also which half) is added multiple times to fool the adversary.

The advantage of this operation is that no set of n or less than n participants can reconstruct the secret and all the participants combined have complete information about the secret. Hence this constitutes a ramp scheme.

Now we divide the shares into $O(\log n)$ blocks of size $O(\log n)$ each and proceed as before. This whole operation can be done in AC^0 as proved later.

1. One can concatenate the shares of the older participants with padding and permutations and give it to the new participant. But this method fails as there is the original group of n participants who can completely reconstruct the secret and hence it would not be a ramp scheme. Hence modifying the shares of the original participants is necessary.
2. During reconstruction, in addition to applying the inverse permutation, the blocks of the participants who were modified have to be restored. This whole process fools the adversary because the adversary only sees a constant fraction of the secret. In this case the overall effect of our modification is an increase in the number of repeated characters combined with random permutations. Hence using a Chernoff type bound one can conclude that the adversary essentially learns nothing. Thus we have an $(n+1, n, k_0)$ robust evolving ramp scheme.
3. When a set of new participants arrive, we can repeat this same process for each of the new participants, until the block size is exhausted. When we cannot accommodate any more new participant, we refresh the whole system. That is, now the we consider all the available participants, old and new, as a set of participants and construct the whole scheme afresh. The new number of participants is n added to the number of new participants that arrived.

Remark 3. One issue here is how the participants A and B are selected. Who makes the decision as to which participants' shares are modified to construct shares of the new participants. We have two ways to go about it.

1. Firstly the dealer can order the participants and include the information multiple times in the shares. When a new participant arrives the old shares are modified according to the order of the old participants. This is helpful in the case of a dealer-free situation and the participants modify their shares according to the order themselves.
2. When the dealer is present the dealer chooses as per the order which shares to modify to construct the new share. As a tradeoff we assume that a little storage is available to keep the information of the order of the shares.
3. In either way the adversary should not be able to get any information from this order. Otherwise, the adversary, using the order of the participants may completely determine the shares of the old and new participants.

3.3 Technical Details

Construction 1: A basic construction for accommodating one new participant We shall assume previous constructions using random permutations, k-wise independent generators and the ones using asymptotically good error correcting codes. We construct the secret sharing scheme $(Share_1, Rec_1)$ as follows. First we recall some notations adopted from Cheng et al. [10].

For any $n, k, m \in \mathbb{N}$ with $k, m \leq n$, alphabets Σ_0, Σ, let $(Share, Rec)$ be an (n, k) 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 [10] with alphabet Σ, message length m_C, where $m_C = \delta_0 n_C$, $k_C = \delta_1 n_C$, $n_C = O(logn)$ for some constants δ_0 and δ_1.

For any constant $a \geq 1$, $\gamma \in (0, 1]$, the paper by Ishai et.al [10] constructs the following $(n_1 = O(n^a), k_1 = \Omega(n_1)$ secret sharing scheme $(Share_1, Rec_1)$ with share alphabet $\Sigma \times [n_1]$, message alphabet Σ, message length $m_1 = \Omega(n_1)$. For clarity we include the algorithm as in [10].

Algorithm : Construction 1

- The $Share_1$ function is as follows :- $Share_1 : \Sigma^{m_1} \rightarrow (\Sigma \times [n_1])^{n_1}$.
  1. *Let* $\bar{n} = \Theta(n^{a-1})$ *with large enough constant factor.*
  2. *(Independent generator step) :- Let* $g_\tau : \Sigma_0^{m\bar{n}} \rightarrow \Sigma^{m_1}$ *be the l-wise independent generator* where $l = \Omega(\frac{m\bar{n}log|\Sigma_0|}{log|\Sigma|})^{1-\gamma}$.
  3. *For a secret* $x \in \Sigma^{m_1}$ *, we draw a string* $r = (r_1, ..., r_{\bar{n}})$ *uniformly from* $\Sigma_0^{m\bar{n}}$.
  4. *Let* $y = (y_s, y_g)$*, where* $y_s = (Share(r_1), ..., Share(r_n)) \in (\Sigma^n)^{\bar{n}}$ *and* $y_g = g_\tau(r) \oplus x \in \Sigma^{m_1}$.
  5. *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* $n_s = \lceil \frac{n}{m_C} \rceil \bar{n}$.
  6. *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* $n_g = \lceil \frac{m1}{m_C} \rceil$.
  7. *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* $n_1 = (n_s + n_g)n_C$.
  8. *(Generate a random permutation)* $\pi : [n_1] \rightarrow [n_1]$ *apply it on* $y1$ *and this is the output.*
- At this stage a new participant arrives. Denote the new participant by t. Let us suppose that the new participant has obtained a share by construction 3 in section 3.5. We denote it by $S(t)$. From our discussion first want to add the participant to a suitable block, and proceed. But in step 8 of the previous algorithm we have already applied a a random permutation. Hence we can do two things,

1. We can embed $S(t)$ in $y1$(i.e., adding elements to the string $y1$ at suitable positions) according to the random permutation already applied(this is helpful when the dealer is present as the new participant is added)
2. We can concatenate $S(t)$ with $y1$ from step 8 of previous algorithm and apply one more random permutation to the concatenated string to get the output. (This helps in the scenario when the dealer is absent, and some participant(s) have the authority to distribute shares among the new participants.)

In both the cases we store the relevant information multiple times.

Algorithm : Case 1 - Dealer is present

1. *Add elements of the string $S(t)$ at suitable positions of the string $y1$ as per the random permutation.*
2. *Output is the new string y_t with the share of the new participant added.*

Algorithm : Case 2 - Dealer is absent

1. *Concatenate $y1$ and $S(t)$ to get the string y_t^*.*
2. *Store the relevant information multiple times.*
3. *Apply a random permutation on the elements of the string y_t^* to get the output y_t.*

Reconstruction : The reconstruction function Rec_1 is as follows :

Algorithm : Case 1 - Dealer was present

1. *Retrieve $S(t)$ from y_t.*
2. *Using $S(t)$, restore the original shares of the corresponding old participants.*
3. *Delete $S(t)$.*
4. *Compute the inverse permutation 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 $g_\tau(r) \oplus y_g$.*

Algorithm : Case 2 - Dealer was absent

1. *Compute the inverse permutation to get y_t^*.*
2. *Using $S(t)$, restore the original shares of the corresponding old participants.*
3. *Delete $S(t)$ from y.*
4. *Compute the inverse permutation 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 $g_\tau(r) \oplus y_g$.*

Construction 2: Accommodating a collection of new participants Algorithm : Construction 2

1. *The function $Share_1$ is same in this case.*
2. *A new bounded collection of elements arrive.*
3. *Repeat steps of the previous construction (when a single participants) for each element in the new collection keeping the bounds so that the operations can be done in AC^0.*
4. *Refresh when the blocks are exhausted.*
5. *Rec_1 is same as in the previous case. Apply it for each new participant.*

Theorem 2. *$Share_1$ and Rec_1 can also be computed by AC^0 circuits.*

Proof. From the paper by Cheng et.al [10], 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 of the new participant.(This can be done is AC^0 as discussed later)
2. Concatenating the share $S(t)$ to $y1$. This can be done in AC^0.
3. Applying a random permutation which is in AC^0.

As was shown in [10], the reconstruction function can be computed in AC^0. In our case the additional functions we are computing are

1. Inverse permutation 2.
2. Restoring the original shares of the old participants.
3. Deleting the shares of the old participants.

Now the inverse permutation can be computed in AC^0. As we shall see later that restoring the share includes dividing a share into two halves and concatenating to the half of another share. 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. *If the reconstruction error of (Share; Rec) is η , then the reconstruction error 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 [10]. Since the proof is same we refer the reader to Lemma 3.4 in [10].

3.4 Privacy

Our scheme differs from that of Cheng et.al[10] when a new participant (or a collection of new participants) arrives. At this stage we recall Remark 3.4.

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 [10].

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.

To show privacy, we need the following Chernoff Bound.

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. *(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 [10].

Lemma 1. *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 2. *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 3. *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 [10]. 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.

Using the above lemmas and Lemma 3.11 of [10] we have the following estimates in our case :

Theorem 5. *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 $(n' = O(n^a logn); (1 - \alpha)n')$ 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.*

This theorem follows from Theorem 3.12 of [10], theorem 3.5, theorem 3.6, remark 3.4, the note in section 3.4 and the section 3.5 of our paper.

Construction 3 : Evolving access structures in AC^0 - Share of the new participant In this case the $Share_1$ function and the distribution to the new participant is same as before. The reconstruction function Rec_1 is probabilistic.

Let us suppose that we have constructed the $Share_1$ function. Suppose at this stage a new participant arrives. We denote the new participants as t. We do the following steps to get the share of t as $S(t)$:

<u>Algorithm : Construction 3</u>

1. *Select a random participant from the old set of participants, say A (this step is mentioned as random as the adversary has no information about the order of the participants.).*
2. *Denote the share of A as $\{A_1, A_2\}$. Here A_1 denotes the first half of the share string of A and A_2 denotes the second half of the share string of A.*
3. *Take the first half of the share of A, i.e., A_1 and copy it as the first half of the share of t.*
4. *Choose another old participant(next in order) from the remaining participants, say B.*

5. *Denote the share of* B *as* $\{B_1, B_2\}$*. Here* B_1 *denotes the first half of the share string of* B *and* B_2 *denotes the second half of the share string of* B*.*
6. *The new share of* A is $\{B_1, A_2\}$.
7. *The share of* t is $\{A_1, B_1\}$.

For the reconstruction procedure , it is clear that using the reconstruction function as before, all the $n+1$ participants together can reconstruct the secret. But now if n out of the $n+1$ participants are chosen, they do not have complete information about the secret.

Note : Here we note that when a collection of new participants arrive we need to choose new shares in a manner so that duplicate shares are not distributed among new participants. This can be done by choosing the unused partitions of shares of old participants or the shares of a different set of old participants can be modified for each new participant.

Theorem 6. *$S(t)$ in construction 3 can be computed in AC^0.*

Proof. Since copying and concatenating strings can be done in AC^0, $S(t)$ can be computed in AC^0.

3.5 Estimates and Bounds

Share size of a new participant

- **When a new participant arrives :** When a new participant arrives, we choose two participants of the previous generation. Take one half of the share of the first participant, and one half of the share of the second participant, concatenating them , applying a random permutation which is the share of the new participant. So in this case, essentially we get that the size of the share for the new participant remains the same.
- **When a group of participant arrives :** In this case the steps as before are repeated for each new participant and the share size is the same as before. The sizes of the share changes only when we need to refresh the whole system, i.e., when the blocks are totally exhausted. When we refresh the system and redistribute the shares, the size of the shares is exponential in the total number of participants.

Total number of participants that can be accommodated

- The size of the outer and the inner levels are $O(\log^2 n)$ and $O(\log n)$ respectively. For the computation to be carried out in AC^0, we need to keep the size of the outer block $O(\log^2 n)$. Hence we can accommodate upto $O(\log^2 n)$ new participants for each block.

Bounds on the number of shares to be modified As mentioned in the previous section when a new participant arrives we are modifying the share of one old participant. The points to note for further analysis are :

1. *The shares are divided into two halves.*
2. *Only the first half of the share of A is modified.*
3. *Only the share of A is modified.*

Each of these points can be extended as we see.

1. Instead of dividing the shares into two halves, we may divide them into 3 equal sized blocks. Say $A = \{A_1, A_2, A_3\}$ and $B = \{B_1, B_2, B_3\}$. Following the method in section 3.5, we can modify the shares of A as $\{B_1, A_2, A_3\}$ and give the share of t as $\{A_1, B_2, B_3\}$ or $\{A_1, B_2, A_3\}$ and so on. Extending this idea, we can divide the shares into more partitions of smaller length. Ours is a code based secret sharing scheme and hence we cannot divide the shares into arbitrarily small partitions. This is because suppose we divide a share into very small partitions and extend the method mentioned above to modify the share of one participant (say A) and give it to the new participant (say t), it may happen that the share of A remains the same which is not desirable as we want to construct a ramp scheme and in this case n participants can reconstruct the secret. If d is the distance of the code, we can divide the share into at most d partitions.
2. Instead of modifying the first half of the share, we can modify the second half of the share. The share of t can also be $\{B_1, A_1\}$. Hence the number of ways we can give shares to the new participant is 8. This increases as we increase the number of partitions. The advantage is that we can accommodate more new participants without modifying any other old participant.
3. In our construction only the share of A was modified using the share of B. Selecting A and B was done by a predefined order on the participants. Along the line of *sequential secret sharing* one can give an hierarchical structure on this scheme. On the set of participants let us suppose that the dealer assigns a set of participants to have a higher hierarchy (as per the predefined order), i.e., a subset of participants (say P_1) only whose share are to be modified and are given to the new participants. This is done until the shares of the participants in P_1 cannot be modified anymore. Then we look at another subset of participants (say P_2) which is disjoint from P_1. The condition for P_1 to be disjoint from P_2 is not necessary but we consider it for simplicity. Hence in one generation(i.e., when new participants are accommodated before exhaustion) we have the following hierarchy :

   Dividing participants into groups of different levels.

   (a) *Dealer D and n participants P.*
   (b) *The dealer puts an order on the participants.*
   (c) *The dealer shares the secret among the participants.*
   (d) *The set of participants P is divided into subsets $P_1, P_2, ..$ and so on.*

(e) *When a new participant (say t_1) arrives, modify the shares of P_1 and give the share to t_1.*
(f) *Continue this process until the shares of the participants of P_1 cannot be modified anymore.*
(g) *When P_1 is exhausted, continue the same process with P_2 and so on.*
(h) *When all the subsets of the old set of participants are exhausted, i.e., the system cannot accommodate any more new participants the whole system is refreshed. The dealer again shares a secret among all the available participants.*

Conclusion and Open Issues

We give an explicit construction of an evolving secret sharing scheme implementable in AC^0. Moreover, we consider the problem of redistributing secret shares (in AC^0) in absence of dealer. On the downside, our scheme can only accommodate a bounded number of new participants for redistributing the secret shares. Possibility (or impossibility) of constructing more threshold schemes in AC^0 can be some of the challenging questions. Moreover, fine-grained analysis of secret sharing schemes in AC^0 or NC^1 can be an interesting follow up work.

Acknowledgement

This research work of S. Dutta was financially supported by National Institute of Information and Communications Technology (NICT), Japan under the NICT International Invitation Program.

References

1. Akavia, A., Bogdanov, A., Guo, S., Kamath, A., Rosen, A.: Candidate weak pseudorandom functions in ac^0 9675 mod$_2$. In: Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014. pp. 251–260 (2014), `https://doi.org/10.1145/2554797.2554821`
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in nc^0. SIAM J. Comput. 36(4), 845–888 (2006), `https://doi.org/10.1137/S0097539705446950`
3. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography with constant input locality. J. Cryptology 22(4), 429–469 (2009), `https://doi.org/10.1007/s00145-009-9039-0`
4. Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Average-case fine-grained hardness. Electronic Colloquium on Computational Complexity (ECCC) 24, 39 (2017), `https://eccc.weizmann.ac.il/report/2017/039`
5. Beimel, A., Othman, H.: Evolving ramp secret-sharing schemes. In: Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings. pp. 313–332 (2018), `https://doi.org/10.1007/978-3-319-98113-0\_17`
6. Blakley, G.R.: Safeguarding cryptographic keys. In: AFIPS 1979. pp. 313–317 (1997)

7. Bogdanov, A., Ishai, Y., Viola, E., Williamson, C.: Bounded indistinguishability and the complexity of recovering secrets. In: Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III. pp. 593–618 (2016), `https://doi.org/10.1007/978-3-662-53015-3\_21`
8. Campanelli, M., Gennaro, R.: Fine-grained secure computation. In: Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part II. pp. 66–97 (2018), `https://doi.org/10.1007/978-3-030-03810-6\_3`
9. Chen, L., Gollmann, D., Mitchell, C.J.: Key escrow in mutually mistrusting domains. In: Security Protocols, International Workshop, Cambridge, United Kingdom, April 10-12, 1996, Proceedings. pp. 139–153 (1996), `https://doi.org/10.1007/3-540-62494-5\_14`
10. Cheng, K., Ishai, Y., Li, X.: Near-optimal secret sharing and error correcting codes in \mathsf acˆ0 AC 0. In: Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II. pp. 424–458 (2017), `https://doi.org/10.1007/978-3-319-70503-3\_14`
11. Cheraghchi, M.: Nearly optimal robust secret sharing. In: IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016. pp. 2509–2513 (2016), `https://doi.org/10.1109/ISIT.2016.7541751`
12. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: 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. pp. 471–488 (2008), `https://doi.org/10.1007/978-3-540-78967-3\_27`
13. Degwekar, A., Vaikuntanathan, V., Vasudevan, P.N.: 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. pp. 533–562 (2016), `https://doi.org/10.1007/978-3-662-53015-3\_19`
14. Desmedt, Y., Jajodia, S.: 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)
15. Desmedt, Y., Morozov, K.: Parity check based redistribution of secret shares. In: IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015. pp. 959–963 (2015), `https://doi.org/10.1109/ISIT.2015.7282597`
16. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: 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. pp. 384–393 (1997), `https://doi.org/10.1109/SFCS.1997.646127`
17. Guruswami, V., Smith, A.D.: Optimal rate code constructions for computationally simple channels. J. ACM 63(4), 35:1–35:37 (2016), `https://doi.org/10.1145/2936015`
18. Ito, M., Saio, A., Nishizeki, T.: Multiple assignment scheme for sharing secret. J. Cryptology 6(1), 15–20 (1993), `https://doi.org/10.1007/BF02620229`
19. Karchmer, M., Wigderson, A.: On span programs. In: Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993. pp. 102–111 (1993), `https://doi.org/10.1109/SCT.1993.336536`

20. Komargodski, I., Naor, M., Yogev, E.: How to share a secret, infinitely. In: Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. pp. 485–514 (2016), https://doi.org/10.1007/978-3-662-53644-5_19
21. Komargodski, I., Paskin-Cherniavsky, A.: Evolving secret sharing: Dynamic thresholds and robustness. In: Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II. pp. 379–393 (2017), https://doi.org/10.1007/978-3-319-70503-3_12
22. Kurihara, J., Kiyomoto, S., Fukushima, K., Tanaka, T.: On a fast (k, n)-threshold secret sharing scheme. IEICE Transactions 91-A(9), 2365–2378 (2008), https://doi.org/10.1093/ietfec/e91-a.9.2365
23. Lai, R.W.F., Malavolta, G., Schröder, D.: 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. pp. 279–309 (2018), https://doi.org/10.1007/978-3-030-03332-3_11
24. Minsky, M., Papert, S.: Perceptrons. MIT Press (1969)
25. Naor, M., Shamir, A.: Visual cryptography. In: Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings. pp. 1–12 (1994), https://doi.org/10.1007/BFb0053419
26. Nojoumian, M., Stinson, D.R.: On dealer-free dynamic threshold schemes. Adv. in Math. of Comm. 7(1), 39–56 (2013), https://doi.org/10.3934/amc.2013.7.39
27. Nojoumian, M., Stinson, D.R.: Sequential secret sharing as a new hierarchical access structure. J. Internet Serv. Inf. Secur. 5(2), 24–32 (2015), http://isyou.info/jisis/vol5/no2/jisis-2015-vol5-no2-02.pdf
28. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979), http://doi.acm.org/10.1145/359168.359176
29. Stinson, D.R., Wei, R.: 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. pp. 200–214 (1999), https://doi.org/10.1007/3-540-46513-8_15