Electronic Colloquium on Computational Complexity, Revision 4 of Report No. 23 (2016)

How to Share a Secret, Infinitely*†

Ilan Komargodski Moni Naor Eylon Yogev

Abstract

Secret sharing schemes allow a dealer to distribute a secret piece of information among several parties such that only qualified subsets of parties can reconstruct the secret. The collection of qualified subsets is called an access structure. The best known example is the k-threshold access structure, where the qualified subsets are those of size at least k. When $k = 2$ and there are n parties, there are schemes for sharing an ℓ-bit secret in which the share size of each party is roughly $\max\{\ell, \log n\}$ bits, and this is tight even for secrets of 1 bit. In these schemes, the number of parties n must be given in advance to the dealer.

In this work we consider the case where the set of parties is not known in advance and could potentially be infinite. Our goal is to give the t^{th} party arriving the smallest possible share as a function of t. Our main result is such a scheme for the k-threshold access structure and 1-bit secrets where the share size of party t is $(k-1) \cdot \log t + \mathsf{poly}(k) \cdot o(\log t)$. For $k = 2$ we observe an *equivalence* to prefix codes and present matching upper and lower bounds of the form $\log t + \log\log t + \log\log\log t + O(1)$. Finally, we show that for any access structure there exists such a secret sharing scheme with shares of size 2^{t-1}.

*A preliminary version of this paper appeared in the 14th International Conference on Theory of Cryptography (TCC 2016-B) [KNY16].

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science Israel, Rehovot 76100, Israel. Email: {ilan.komargodski,moni.naor,eylon.yogev}@weizmann.ac.il. Research supported in part by grants from the Israel Science Foundation (grants no. 1255/12 and 950/16), BSF and from the I-CORE Program of the Planning and Budgeting Committee and the Israel Science Foundation (grant no. 4/11). Moni Naor is the incumbent of the Judith Kleeman Professorial Chair. Ilan Komargodski is supported in part by a Levzion fellowship.

ISSN 1433-8092

1 Introduction

640K ought to be enough for anybody

Misattributed to Bill Gates, 1981

Secret sharing is a method by which a secret piece of information can be distributed among n parties so that any qualified subset of parties can reconstruct the secret, while every unqualified subset of parties learns nothing about the secret. The collection of qualified subsets is called an access structure. Secret sharing schemes are a basic primitive and have found applications in cryptography and distributed computing; see the extensive survey of Beimel [Bei11]. A significant goal in secret sharing is to minimize the share size, namely, the amount of information distributed to the parties.

Secret sharing schemes were introduced in the late 1970s by Shamir [Sha79] and Blakley [Bla79] for the k-out-of-n threshold access structures that includes all subsets of cardinality at least k for $1 \leq k \leq n$. Their constructions are fairly efficient both in the size of the shares and in the computation required for sharing and reconstruction. Ito, Saito, and Nishizeki [ISN93] showed the existence of a secret sharing scheme for every (monotone) access structure. In their scheme the size of the shares is proportional to the depth 2 complexity of the access structure when viewed as a Boolean function (and hence shares are exponential for most structures). Benaloh and Leichter [BL88] gave a scheme with share size polynomial in the monotone formula complexity of the access structure. Karchmer and Wigderson [KW93] generalized this construction so that the size is polynomial in the monotone span program complexity.

All of these schemes require that an upper bound on the number of participants is known in advance. However, in many scenarios this is either unrealistic or prone to disaster. Moreover, even if a crude upper bound n is known in advance, it is preferable to have shares as small as possible if the eventual number of participants is much smaller than this bound on n.

In this work we consider the well motivated, yet almost unexplored[1], case where the set of parties is *not* known in advanced and could potentially be infinite. Our goal is to give the t^{th} party arriving the smallest possible share as a function of t. We require that in each round, as a new party arrives, there is no communication to the parties that have already received shares, i.e. the dealer distributes a share only to the new party. We call such access structures evolving: the parties arrive one by one and, in the most general case, a qualified subset is revealed to the dealer only when all parties in that subset are present (in special cases the dealer knows the access structure to begin with, just does not have an upper bound on the number of parties). For this to make sense, we assume that the changes to the access structure are monotone, namely, parties are only added and qualified sets remain qualified.

Our first result is a construction of a secret sharing scheme for *any* evolving access structure.[2]

Theorem 1.1. *For every evolving access structure there is a secret sharing scheme for a 1-bit secret where the share size of the t^{th} party is 2^{t-1}.*

Then, we construct more efficient schemes for specific access structures. We focus on the evolving k-threshold access structure for $k \in \mathbb{N}$, where at any point in time any k parties can reconstruct the secret but no $k-1$ parties can learn anything about the secret.

[1] But see the work of Csirmaz and Tardos [CT12] discussed below.

[2] A comparison with the work of Csirmaz and Tardos [CT12] is given below in Section 1.2.

Theorem 1.2. *For every $k, \ell \in \mathbb{N}$, there is a secret sharing scheme for the evolving k-threshold access structure and an ℓ-bit secret in which for every $t \in \mathbb{N}$ the share size of the t^{th} party is $(k-1) \cdot \log t + \mathsf{poly}(k, \ell) \cdot o(\log t)$.*

For $k = 2$, we present a tight connection to prefix codes for the integers. A prefix code is a code in which no codeword is a prefix of any other codeword. These codes are widely used, for example in country calling codes, the UTF-8 system for encoding Unicode characters, and more.

Theorem 1.3. *Let $\sigma \colon \mathbb{N} \to \mathbb{N}$. A prefix code for the integers in which the length of the t^{th} codeword is $\sigma(t)$ exists if and only if a secret sharing scheme for the evolving 2-threshold access structure and 1-bit secret in which the share size of the t^{th} party is $\sigma(t)$.*

As a corollary, we get that there is a secret sharing scheme for the evolving 2-threshold access structure and a 1-bit secret in which the share size of the t^{th} party is $\log t + 2 \log \log t + 2$. Moreover, this is optimal: there is *no* secret sharing scheme for this access structure in which the share size of the t^{th} party is bounded by $\log t + \log \log t + O(1)$ bits. See Corollaries 4.2 and 4.3 for the precise statements.

1.1 Discussion

Schemes for general access structures. In the classical setting of secret sharing many schemes are known for general access structures, depending on their representation [ISN93, BL88, KW93]. All of these schemes result with shares of exponential size for general access structures. One of the most important open problems in the area of secret sharing is to prove the necessity of long shares, namely, find an access structure (even a non-explicit one) that requires exponential size shares.

Our scheme for general evolving access structures also results with exponential size shares. Since any access structure can be made evolving, we cannot hope to obtain anything better than exponential in general (unless we have a major breakthrough in the classical setting).

Threshold schemes. In the classical setting there are several different schemes for the threshold access structure. One of the best such schemes (in terms of the computation needed for sharing and reconstruction and in terms of the share size) is due to Shamir [Sha79]. In this scheme, to share a 1-bit secret among n parties, roughly $\log n$ bits have to be distributed to each party. It is known that $\log n$ bits are essentially required, so Shamir's scheme is optimal (see [CCX13] for the original proof of Kilian and Nisan [KN90], an improvement, and a discussion of the history; see also [BGK16]).

Let us review Shamir's scheme for the k-out-of-n threshold access structure. The dealer holding a secret bit s, samples a random polynomial $p(\cdot)$ of degree $k-1$ with coefficients over $\mathsf{GF}(q)$, where the free coefficient is fixed to be s, and gives party $i \in [n]$ the field element $p(i)$. The field size, q, is chosen to be the smallest prime (or a power of a prime) larger than n. Correctness of the scheme follows by the fact that k points on a polynomial of degree $k-1$ completely define the polynomial and allow for computing $p(0) = s$. Security follows by a counting argument showing that given less than k points, both possibilities for the free coefficient are equally likely. The share of each party is an element in the field $\mathsf{GF}(q)$ that can be represented using $\log q \approx \log n$ bits. Notice that the share size is independent of k.

As a first attempt one might try to adapt this procedure to the *evolving* setting. But since n is not fixed, what q should we choose? A natural idea is to use an extension field. Roughly, we

would simulate the dealer for Shamir's scheme, sample a random polynomial of degree $k-1$ and increase the field size from which we compute shares as more parties arrive. Ideally, for the share of the t^{th} party we will use a field of size $O(t)$. This implies that the share size of party t would be $\log(O(t)) \ll \log t + \log\log t$ for large enough t. The lower bound in Corollary 4.3 means that *no such solution can work*!

We take a different path for obtaining efficient schemes. For example, for $k = 2$, our scheme results with optimal share size (up to additive lower order terms) for the t^{th} party: the first term is $\log t$ (without hidden constant factors) and there is an additional lower order term that depends on $\log\log t$. See Section 4 for details.

Linearity of our schemes. In a linear scheme the secret is viewed as an element of a finite field, and the shares are obtained by applying a linear mapping to the secret and several independent random field elements. Equivalently, a linear scheme is defined by requiring that each qualified set reconstructs the secret by applying a linear function to its shares [Bei96, Section 4.1]. Most of the known schemes are linear (see [BI01] for an exception). Linear schemes are very useful for updating and manipulating secret shares (cf. proactive secret sharing [HJKY95]) and have many applications, most notably for secure multi-party computation [BGW88, CDM00]. Our schemes from Theorems 1.1 and 1.2 are linear, whereas the scheme based on prefix codes from Theorem 1.3 is non-linear.

1.2 Related Work

The work of Csirmaz and Tardos. Most similar to our setting is the notion of *on-line* secret sharing of Csirmaz and Tardos [CT12]. Csirmaz and Tardos present a scheme for any evolving access structure with the restriction that every party participates in at most d qualified sets, where d is an upper bound known in advance. The share size of every party in their scheme is linear in d. In comparison, Theorem 1.1 works without any such restriction but the share size is larger. In addition, Csirmaz and Tardos presented a scheme for the evolving 2-threshold access structure in which the share size of party t is linear in t. We use this scheme as a base for our 2-threshold scheme (Theorem 1.2) which gives an exponential improvement on their scheme.

Other "evolving" settings. There are numerous areas where systems are designed to work without any fixed upper bound on the size or the duration they will be used. A few examples include prefix codes of the integers (a.k.a. prefix-free encodings), such as the Elias code [Eli75] or the online encoding of Dodis et al. [DPT10], labeling nodes for testing adjacency in possibly infinite graphs [KNR92], forward-secure signatures with an unbounded number of time periods [MMM02], and data structures for approximate set membership (Bloom filters) for sets of unknown size [PSW13].

Other connections to prefix codes. There are other setting in which problems were shown to be tightly related to prefix codes. For example, Bentley and Yao [BY76] studied the *unbounded search* problem (i.e. the problem of comparison-based search in an array of unbounded size) and showed an efficient deterministic algorithm for the problem. Moreover, they observed that every such algorithm implies a prefix-free encoding of the integers. Another work with related ideas was of Even and Rodeh [ER78] who showed a method for inserting delimiters in a contiguous sequence of strings without adding new symbols and with minimal overhead.

1.3 Overview of Our Constructions and Techniques

First, we overview our construction for general evolving access structures. Then, we describe our construction for the evolving 2-threshold access structure. This serves as a warm-up for our more general construction for k-threshold access structures. Lastly, we discuss the connection with prefix codes.

General evolving access structures. Let $\mathcal{A}$ be any evolving access structure and let $\mathcal{A}_t = \mathcal{A} \cap [t]$ be the qualified sets at time t. Note that the dealer does not know $\mathcal{A}$ in advance but is only given $\mathcal{A}_t$ when the t^{th} party arrives. Let $s \in \{0,1\}$ be the secret to be shared. The share of party $t \in \mathbb{N}$ consists of 2^{t-1} bits, each denoted by r_A for $A \subseteq [t-1]$. The r_A's are generated as follows: if party t "completes" a minimal qualified set with $A = \{i_1, \ldots, i_k\}$, namely $A \cup \{t\} \in \mathcal{A}_t$, then the dealer gives party t the bit $r_{A\cup\{t\}} = r_{\{i_1\}} \oplus \ldots \oplus r_{\{i_1,\ldots,i_k\}} \oplus s$. Otherwise, if $A \cup \{t\}$ is unqualified, then the dealer sets $r_{A\cup\{t\}} \leftarrow \{0,1\}$ to be a uniformly random bit. Thus, by XORing the appropriate shares a qualified set can recover s. Details can be found in Section 3.

Prefix codes and evolving 2-threshold. Given any prefix code in which the length of the t^{th} codeword is $\sigma(t)$, we construct a secret sharing scheme for the evolving 2-threshold access structure in which the share size of the t^{th} party is $\sigma(t)$. The dealer maintains an infinitely (random) growing string w and gives the t^{th} party arriving either the string $w[1:\sigma(t)]$ (if the secret to share is 0), or $w[1:\sigma(t)] \oplus \Sigma(t)$, where $w[1:\sigma(t)]$ is the $\sigma(t)$-bit length prefix of w and $\Sigma(t)$ is the encoding of the number t using the prefix code. One can verify that each single share does not leak any information about the secret while any two parties can together decide if one's share is a prefix of the other's and thus recover the secret. Using Elias's prefix code constructions [Eli75] we get a scheme in which the share size is roughly $\log t + 2\log\log t$ bits.

On the other hand, any secret sharing scheme for the evolving 2-threshold access structure in which the share size of the t^{th} party is $\sigma(t)$, implies the existence of a prefix code in which the length of the t^{th} codeword is $\sigma(t)$. This comes from the fact that the share lengths in a 2-threshold scheme must satisfy Kraft's inequality which is a sufficient condition to yield prefix codes.

Evolving 2-threshold, directly. The scheme that results from the connection to prefix codes is optimal when sharing a 1-bit secret. Our next step is to consider a *direct* construction of a scheme for this access structure which has several advantages: (1) it supports sharing long secrets much more efficiently, (2) it serves as a warm-up to our construction for the evolving k-threshold access structure, and (3) it is linear (a property that may be important in applications). We focus on sharing a single bit in this overview and refer to Section 4.2 for the general case.

The approach of [CT12] for the evolving 2-threshold access structure is to give party t a random bit b_t and all bits $s \oplus b_1, \ldots, s \oplus b_{t-1}$. This clearly allows for each pair of parties to reconstruct the secret and ensures that for every single party the secret remains hidden. The share size of the t^{th} party in this scheme is t. (Essentially the same scheme also follows from our general construction in Section 3 with a simple efficiency improvement described towards the end of that section.) Generalizing this idea to larger values of k results with shares of size roughly t^{k-1}.

Whereas the above approach is somewhat naive (and very inefficient in terms of share size), our construction is more subtle and results with exponentially shorter shares. Our main building block is a *domain reduction* technique which allows us to start with a naive solution and apply it only on a small number of parties to get an overall improved construction. Details follow.

We assign each party a generation, where the g^{th} generation consists of 2^g parties (i.e. the generations are of geometrically increasing size). Within each generation we execute a standard

secret sharing scheme for 2-threshold. Notice that here we know exactly how many parties are in the same generation: party t is part of generation $g = \lfloor \log t \rfloor$ and the size of that generation is $\text{SIZE}(g) \leq t$. A standard secret sharing scheme for 2-out-of-t costs roughly $\log t$ bits (using Shamir's scheme; see Claim 2.4). This solves the case in which both parties come from the same generation.

To handle the case where the two parties come from different generations we use a (possibly naive) scheme for the *evolving* 2-threshold access structure. For each generation we generate *one* share for the evolving scheme and give it to each party in that generation. Thus, if two parties from different generations come together they hold two different shares for the evolving scheme that allow them to reconstruct the secret. Since we generate one share of the evolving scheme per generation, party t holds the share of the $(g = \log t)^{\text{th}}$ party of the evolving scheme!

Summing up, if we start with a scheme in which the share size of the t^{th} party is $\sigma(t)$, then we end up with a scheme with share size roughly $\sigma'(t) = \log t + \sigma(\log t)$. To get our result we start with a scheme in which $\sigma(t) = t$ (described above) and iteratively apply this argument to get better and better schemes.

Evolving k-threshold. There are several ideas underlying the generalization of the 2-threshold scheme to work for any threshold k. As before, we assign each party to a generation, but now the g^{th} generation is roughly of size $2^{(k-1)\cdot g}$. This means that party t is in generation $g = \lfloor (\log t)/(k-1) \rfloor$ that includes $\text{SIZE}(g) = t \cdot 2^{k-1}$ parties. Again, within a generation we apply a standard k-out-of-$\text{SIZE}(g)$ secret sharing scheme. This costs us $\log(\text{SIZE}(g)) \leq \log t + k$ bits using Shamir's scheme. This solves the problem if k parties come from the same generation.

We are left with the case where the k parties come from at least two different generations. For this we use a (possibly naive) scheme for the *evolving* k-threshold access structure. For each generation we generate $k-1$ shares $s_1, \ldots, s_{k-1}$ for the evolving scheme and share each s_i using a standard i-out-of-$\text{SIZE}(g)$ secret sharing scheme. Thus, if $w \leq k-1$ parties from some generation come together, they can reconstruct $s_1, \ldots, s_w$ which are w shares for the evolving scheme.

Therefore, any k parties (that come from at least two generations) can reconstruct k shares for the evolving k-threshold scheme that enable them to reconstruct the secret. Since we generate $k-1$ shares of the evolving scheme per generation, party t holds (roughly) the share of the $(\log t + k)^{\text{th}}$ party of the evolving scheme.

The share size needed to share each s_i is $\max\{\log(\text{SIZE}(g)), |s_i|\} \leq \max\{\log t + k, \sigma(\log t + k)\}$ (using Shamir's scheme; see Claim 2.4). Summing up, if we start with a scheme in which the share size of the t^{th} party is $\sigma(t)$, then we end up with a scheme with share size roughly $\sigma'(t) = \log t + (k-1) \cdot \max\{\log t + k, \sigma(\log t + k)\}$. A small optimization is that sharing s_1 costs just $|s_1|$, as we can give s_1 to each party (similarly to what we did in the $k = 2$ case).

We want to iteratively apply this domain reduction procedure. For this we have to specify the initial scheme. If we start with the scheme that results from the construction in Theorem 1.1 which has share size roughly 2^t (or roughly t^k with a minor optimization), then the resulting scheme will have a factor that depends *exponentially* on k. This makes the scheme impractical even for small values of k.

To get around this we present a tailor-made construction for the evolving k-threshold in which the share size of party t has *almost linear* dependence on t and k. Specifically, the share size in this scheme is $kt \cdot \log(kt)$. For this, we use a variant of the scheme for general access structures such that we group parties into generations and use the fact that we only care about the number of parties in each generation and not their identities. We associate with each generation g bits r_A for $A = (c_0, \ldots, c_g) \in \{0, \ldots, k\}^g$, where each c_i indicates how many parties arrive from generation i. For

each such A the dealer chooses the element r_A such that if $c_g = 0$, then $r_A = 0$, if $\sum_{i=0}^{g} c_i < k$, the bit r_A is chosen uniformly at random, and if $\sum_{i=0}^{g} c_i = k$, it is set to be $r_A = r_{(c_0)} \oplus \ldots \oplus r_{(c_0,\ldots,c_{g-1})} \oplus s$. If $c_g > 0$, the dealer shares r_A among the parties in generation g using a Shamir c_g-out-of-$\text{SIZE}(g)$ scheme, where $\text{SIZE}(g)$ is the number of parties in a generation. Letting the number of parties in generation g be roughly k^{g+1}, we get that the share size of a party in generation g is $k^{g+1} \cdot \log(k^{g+1})$ (since we apply Shamir's scheme on each r_A). Party t is part of generation $g = \log_k t$ which implies that its share size is $kt \cdot \log(kt)$, as required.

2 Model and Definitions

For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. We denote by log the base 2 logarithm and assume that $\log 0 = 0$. For a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$

We start by briefly recalling the standard setting of (perfect) secret sharing. Let $\mathcal{P}_n = \{1, \ldots, n\}$ be a set of n parties. A collection of subsets $\mathcal{A} \subseteq 2^{\mathcal{P}_n}$ is monotone if for every $B \in \mathcal{A}$, and $B \subseteq C$ it holds that $C \in \mathcal{A}$.

Definition 2.1 (Access structure)**.** An access structure $\mathcal{A} \subseteq 2^{\mathcal{P}_n}$ is a monotone collection of non-empty subsets. Subsets in $\mathcal{A}$ are called qualified and subsets not in $\mathcal{A}$ are called unqualified.

Definition 2.2 (Threshold access structure)**.** For every $n \in \mathbb{N}$ and $1 \leq k \leq n$, let (k, n)-THR be the threshold access structure over n parties which contains all subsets of size at least k.

A (standard) secret sharing scheme involves a dealer who has a secret, a set of n parties, and an access structure $\mathcal{A}$. A secret sharing scheme for $\mathcal{A}$ is a method by which the dealer distributes shares to the parties such that any subset in $\mathcal{A}$ can reconstruct the secret from its shares, while any subset not in $\mathcal{A}$ cannot reveal any information on the secret.

More precisely, a secret sharing scheme for an access structure $\mathcal{A}$ consists of a pair of probabilistic algorithms $(\mathsf{SHARE}, \mathsf{RECON})$. SHARE gets as input a secret s (from a domain of secrets S) and a number n, and generates n shares $\Pi_1^{(s)}, \ldots, \Pi_n^{(s)}$. RECON gets as input the shares of a subset B and outputs a string. The requirements are:

1. For every secret $s \in S$ and every qualified set $B \in \mathcal{A}$, it holds that $\mathbf{Pr}[\mathsf{RECON}(\{\Pi_i^{(s)}\}_{i \in B}, B) = s] = 1$.
2. For every unqualified set $B \notin \mathcal{A}$ and every two different secrets $s_1, s_2 \in S$, it holds that the distributions $(\{\Pi_i^{(s_1)}\}_{i \in B})$ and $(\{\Pi_i^{(s_2)}\}_{i \in B})$ are identical.

The share size of a scheme is the maximum number of bits each party holds in the worst case over all parties and all secrets.

A useful fact that we use in some of our proofs is that concatenation of independently generated shares does not harm security. That is, for any two secrets s_0, s_1 and any two access structures $\mathcal{A}_0, \mathcal{A}_1$, if a set of parties is unqualified in both structures, then it cannot gain any information regarding the secrets. We state this fact for two access structures and two secrets next, but it naturally generalizes to more.

Fact 2.3. *Fix two access structures $\mathcal{A}_0, \mathcal{A}_1$ with corresponding secret sharing schemes, four secrets $s_0^{(0)}, s_1^{(0)}, s_0^{(1)}, s_1^{(1)}$, and a set of parties $B \notin \mathcal{A}_1, \mathcal{A}_2$ that is unqualified both in $\mathcal{A}_0$ and in $\mathcal{A}_1$.*

Then, the distributions $(\{\Pi_{0,i}^{(s_0^{(0)})}, \Pi_{1,i}^{(s_1^{(0)})}\}_{i\in B})$ *and* $(\{\Pi_{0,i}^{(s_0^{(1)})}, \Pi_{1,i}^{(s_1^{(1)})}\}_{i\in B})$ *are identical, where* $\Pi_{c,i}^{(s_b^{(a)})}$ *for* $a, b \in \{0,1\}$ *and* $i \in [n]$ *is a random variable for the share of the* i^{th} *party when sharing the secret* $s_b^{(a)}$ *according to* $\mathcal{A}_b$*.*

The well known scheme of Shamir [Sha79] for the (k,n)-THR access structure (based on polynomial interpolation) satisfies the following.

Claim 2.4 ([Sha79])**.** For every $n \in \mathbb{N}$ and $1 \leq k \leq n$, there is a secret sharing scheme for secrets of length m and the (k,n)-THR access structure in which the share size is ℓ, where $\ell \geq \max\{m, \log q\}$ and $q > n$ is a prime number (or a power of a prime). Moreover, if $k = 1$ or $k = n$, then $\ell = m$.[3]

2.1 Secret Sharing for Evolving Access Structures

We proceed with the definition of an evolving access structure. Roughly speaking, the parties arrive one by one and, in the most general case, a qualified subset is revealed only when all parties in that subset are present (in special cases the access structure is known to begin with, but there is no upper bound on the number of parties). To make sense of sharing a secret with respect to such a sequence of access structures, we require that the changes to the access structure are monotone, namely, parties are only added and qualified sets remain qualified.

Definition 2.5 (Evolving access structure)**.** An evolving access structures $\mathcal{A} \subseteq 2^{\mathbb{N}}$ is a (possibly infinite) monotone collection of subsets of the natural numbers such that for any $t \in \mathbb{N}$, the collection of subsets $\mathcal{A}_t \triangleq \mathcal{A} \cap [t]$ is an access structure.

This definition naturally gives rise to an evolving variant of threshold access structures (see Definition 2.2). Here, we think of k as fixed, namely, independent of the number of parties.

Definition 2.6 (Evolving threshold access structure)**.** For every $k \in \mathbb{N}$, let evolving k-THR be the evolving threshold access structure which contains all subsets of $2^{\mathbb{N}}$ of size at least k.

We generalize the definition of a standard secret sharing scheme to apply for evolving access structures. Intuitively, in this setting, at any point $t \in \mathbb{N}$ in time, there is an access structure $\mathcal{A}_t$ which defines the qualifies and unqualified subsets of parties.

Definition 2.7 (Secret sharing for evolving access structures)**.** Let $\mathcal{A}$ be an evolving access structure. Let S be a domain of secrets, where $|S| \geq 2$. A secret sharing scheme for $\mathcal{A}$ and S consists of a pair of algorithms $(\mathsf{SHARE}, \mathsf{RECON})$. The sharing procedure SHARE and the reconstruction procedure RECON satisfy the following requirements:

1. $\mathsf{SHARE}(s, \{\Pi_1^{(s)}, \ldots, \Pi_{t-1}^{(s)}\})$ gets as input a secret $s \in S$ and the secret shares of parties $1, \ldots, t-1$. It outputs a share for the t^{th} party. For $t \in \mathbb{N}$ and secret shares $\Pi_1^{(s)}, \ldots, \Pi_{t-1}^{(s)}$ generated for parties $\{1, \ldots, t-1\}$, respectively, we let

$$\Pi_t^{(s)} \leftarrow \mathsf{SHARE}(s, \{\Pi_1^{(s)}, \ldots, \Pi_{t-1}^{(s)}\})$$

   be the secret share of party t.

   We abuse notation and sometimes denote by $\Pi_t^{(s)}$ the random variable that corresponds to the secret share of party t generated as above.

[3] Schemes in which the share size is equal to the secret size are known as *ideal* secret sharing schemes.

2. **Correctness:** For every secret $s \in S$ and every $t \in \mathbb{N}$, every qualified subset in $\mathcal{A}_t$ can reconstruct the secret. That is, for $s \in S$, $t \in \mathbb{N}$, and $B \in \mathcal{A}_t$, it holds that

$$\mathbf{Pr}\left[\mathsf{RECON}(\{\Pi_i^{(s)}\}_{i\in B}, B) = s\right] = 1,$$

where the probability is over the randomness of the sharing and reconstruction procedures.

3. **Secrecy:** For every $t \in \mathbb{N}$, every unqualified subset $B \notin \mathcal{A}_t$, and every two secret $s_1, s_2 \in S$, the distribution of the secret shares of parties in B generated with secret s_1 and the distribution of the shares of parties in B generated with secret s_2 are identical. Namely, the distributions $(\{\Pi_i^{(s_1)}\}_{i\in B})$ and $(\{\Pi_i^{(s_2)}\}_{i\in B})$ are identical.

The share size of the t^{th} party in a scheme for an evolving access structure is $\max |\Pi_t|$, namely the number of bits party t holds in the worst case over all secrets and previous assignments.[4]

On choosing the access structure adaptively. One can also consider a stronger definition in which $\mathcal{A}_t$ is chosen at time t (rather than ahead of time) as long as qualified sets remain qualified. In this variant, the RECON procedure gets the additional new qualified sets as an additional parameter. Our construction of a secret sharing scheme for general evolving access structures in Section 3 works for this notion as well.

On the domain of secrets. Unless otherwise stated, we usually assume that the secret is a single bit (either 0 or 1). One can generalize any such scheme to support longer secrets by secret sharing every bit of the secret independently, suffering a multiplicative factor in share size that depends on the length of the secret. When we generalize our schemes to support long secrets, this naive generalization will be our benchmark.

2.2 Warm-Up: Undirected st-Connectivity

We start with a simple warm-up scheme. We show that the standard scheme for the st-connectivity access structure can be easily adapted to the evolving setting. In this access structure parties correspond to edges of an *undirected* graph $G = (V, E)$. There are two fixed vertices in the graph called s and t (where $\mathsf{s}, \mathsf{t} \in V$). A set of parties (i.e. edges) is qualified if and only if they include a path from s to t.

Around 1989 Benaloh and Rudich [BR88] constructed a (standard) secret sharing for this access structure. The dealer, given a secret $s \in \{0,1\}$, assigns with each vertex $v \in V$ a label. For $v = \mathsf{s}$ the label is $w_{\mathsf{s}} = s$, for $v = \mathsf{t}$ the label is $w_{\mathsf{t}} = 0$ and for the rest of the vertices the label is chosen independently uniformly at random $w_v \leftarrow \{0,1\}$. The share of a party $e = (u, v) \in E$ is $w_u \oplus w_v$.

For correctness consider a set of parties that include a path $\mathsf{s} = v_1 v_2 \dots v_k = \mathsf{t}$ from s to t. To reconstruct the secret, the parties XOR their shares to get

$$(w_{v_1} \oplus w_{v_2}) \oplus (w_{v_2} \oplus w_{v_3}) \oplus \cdots \oplus (w_{v_{k-1}} \oplus w_{v_k}) = w_{v_1} \oplus w_{v_k} = s.$$

For security, one can show that any subset of parties that do not form a path from s to t, hold shares which are independent of the secret. One way to prove this is to show that the number of

[4] This means that the share size is bounded, which is almost always the case. An exception is the scheme (for rational secret sharing) of Kol and Naor [KN08] in which the share size does not have a fixed upper bound.

ways to get to the shares of the parties given the secret 0 is equal to the number of ways to get to these shares given the secret 1 (see [Bei11, §3.2]).

One can observe that this access structure and scheme naturally generalize to the evolving setting. In this setting, we consider an evolving (possibly infinite) graph, where the set of nodes and edges are unbounded. At any point in time an arbitrary set of vertices and edges can be added to the graph. An addition of an edge corresponds to a new party added to the scheme. The special vertices s and t are fixed ahead of time and cannot change (this is to ensure the access structure is *evolving*).

Initially, the dealer assigns labels for the special vertices s and t, as before (i.e. it sets $w_{\mathsf{s}} = s$ and $w_{\mathsf{t}} = 0$). For the rest of the vertices the dealer assigns (uniformly random) labels only on demand: When a new edge $e = (u, v)$ is added to the graph (which corresponds to a new party), the dealer gives the party corresponding to the edge e the XOR of the labels of the vertices u and v. Correctness and security of this scheme follow similarly to the correctness and security of the standard scheme. One can see that the share size of each party is exactly the size of the secret.

3 A Scheme for General Evolving Access Structures

We give a construction of a secret sharing scheme for every evolving access structure.

Theorem 3.1 (Theorem 1.1 restated)**.** *For every evolving access structure there is a secret sharing scheme for a 1-bit secret, where for any $t \in \mathbb{N}$, the share size of the t^{th} party is at most 2^{t-1}.*

Proof of Theorem 3.1. Let $\mathcal{A}$ be an evolving access structure. Let $s \in \{0,1\}$ be the secret to be shared. We describe what the dealer stores and how it prepares a share for an arriving party.

At time t the dealer maintains 2^t bits denoted by s_A for all $A \subseteq [t]$. Initially, we set $s_\emptyset = s$. The value of each s_A for $A = \{i_1, \ldots, i_k, t\} \subseteq [t]$ is defined as follows:

1. If $A \notin \mathcal{A}_t$, the dealer samples $r_A \leftarrow \{0,1\}$ uniformly at random and sets $s_A = s_{A\setminus\{t\}} \oplus r_A$. The dealer gives party t the bit r_A.

2. If $A \in \mathcal{A}_t$ and $A \setminus \{t\} \notin \mathcal{A}_{t-1}$, the dealer gives party t the bit $s_{A\setminus\{t\}}$.

Party t holds at most a single bit for every $A \subseteq [t]$ such that $t \in A$. This implies that the share size of party t is at most 2^{t-1} bits.

Correctness and security. We argue correctness at time $t \in \mathbb{N}$, where the access structure is $\mathcal{A}_t$. Let $A = \{i_1, \ldots, i_k, t\}$ be a minimal qualified set of parties present at time t. Since $A \in \mathcal{A}_t$, for every $j \in [k]$, party i_j holds the bit $r_{\{i_1,\ldots,i_j\}}$. Party t, by construction, holds the bit $s_A = r_{\{i_1\}} \oplus \ldots \oplus r_{\{i_1,\ldots,i_k\}} \oplus s$. Therefore, by XOR-ing all the shares the parties in A can recover s.

We prove that our scheme is secure by induction on t. For $t = 1$, the scheme is secure since either the set consisting of first party is qualified, in which case the scheme is secure (since no information is public); if the first party is unqualified by itself, then the this party simply gets a random bit $r_{\{1\}}$ which is independent of the secret. Suppose that the scheme is secure for access structures over $t-1$ parties. Consider a set $A = \{i_1, \ldots, i_k, t\}$ of parties that form an unqualified set relative to $\mathcal{A}_t$.

Consider the dealer right after it performs the sharing to the first party. At this point the dealer maintains two bits: $s_\emptyset$ and $s_{\{1\}}$. Now, we observe that the remaining procedure of the dealer consists of two instances of the scheme for two access structures over $t-1$ parties and two secrets.

The key point is that these secret-sharings are done *independently* (and hence their concatenation is secure). First, the dealer shares the secret $s_\emptyset$ among parties $2, \ldots, t$ w.r.t the access structure $\mathcal{A}_t^0 = \{A \subseteq \{2, \ldots, t\} \mid A \in \mathcal{A}_t\}$ that contains all qualified sets in which the first party does not participate in. Second, the dealer shares the secret $s_{\{1\}}$ among parties $2, \ldots, t$ w.r.t the access structure $\mathcal{A}_t^1 = \{A \subseteq \{2, \ldots, t\} \mid \{1\} \cup A \in \mathcal{A}_t\}$ that contains all qualified sets in which the first party is a member.

Since the original set of parties is unqualified, it must be that (1) the remaining set of parties (i.e. $A \setminus \{1\}$) is unqualified in $\mathcal{A}_t^0$ and (2) either $1 \notin A$ or the remaining set of parties is unqualified in $\mathcal{A}_t^1$. We can apply the induction hypothesis on the sharing according to $\mathcal{A}_t^0$ and get that the secret is independent of the corresponding shares. For the second part, there are two cases: (1) If $1 \notin A$, then the shared secret ($s_{\{1\}}$) is independent of s and therefore the whole view of the shares of A is independent of s. (2) If $1 \in A$ but $A \notin \mathcal{A}_t$, then the set A knows the masking of $s_\emptyset$ which is a random bit $r_{\{1\}}$, however, the induction hypothesis ensures that the shares of the remaining parties are independent of $s_{\{1\}} = s_\emptyset \oplus r_{\{1\}}$, which implies that they are independent of the secret. Each case is independent of one another and in each case the resulting shares are independent of the secret. Using Fact 2.3, we conclude that the secret remains perfectly hidden. ■

The share size in some special cases. In some cases, depending on the access structure, it is possible to analyze the worst case share size more carefully. Recall that in our scheme each party gets two types of bits:

1. A bit for each unqualified subset of $[t]$ that party t participates in. For the case when the access structure is known ahead of time, the only unqualified sets to consider are those that can be expanded to a qualified subset using future parties.

2. A bit for each qualified subset of $[t]$ that party t completes (i.e. it is the last one).

Thus, when the number of unqualified sets is small, we can get a better bound. Consider, for example, the evolving 2-THR access structure. At time t, there are only t unqualified sets (the singletons). Thus, the share size of the t^{th} party is exactly t (we use this fact in Section 4). More generally, for the evolving k-THR access structure, there are $\sum_{i=0}^{k-2} \binom{t-1}{i}$ unqualified sets and $\binom{t-1}{k-1}$ qualified sets which t completes, implying a scheme with share size roughly t^{k-1}.

4 Evolving 2-Threshold

In this section we present an equivalence between prefix codes for the integers and secret sharing schemes for 1-bit secrets and the evolving 2-threshold access structure. This construction gives an efficient scheme (based on Elias's prefix code [Eli75]) which we show to be optimal in terms of share size.

Theorem 4.1 (Theorem 1.3 restated)**.** *Let $\sigma \colon \mathbb{N} \to \mathbb{N}$. A prefix code for the integers in which the length of the t^{th} codeword is $\sigma(t)$ exists if and only if a secret sharing scheme for the evolving 2-threshold access structure and a 1-bit secret in which the share size of the t^{th} party is $\sigma(t)$.*

Corollary 4.2. *There is a secret sharing scheme for the evolving* 2-THR *access structure and a 1-bit secret in which the share size of the t^{th} party is $\log t + 2 \log \log t + 2$.*

Corollary 4.3. *For any constants $c, \ell \in \mathbb{N}$, there is* no *secret sharing scheme for the evolving* 2-THR *access structure in which for every $t \in \mathbb{N}$ the share size of the t^{th} party is at most* $\log t + \log\log t + \ldots + \log^{(\ell)} t + c$, *where* $\log^{(i)}(t)$ *is the i-times repeated* log *of t.*

Furthermore, we give a *direct* construction of a secret sharing scheme the evolving 2-threshold access structure that is more efficient when sharing longer secrets. This scheme has the additional advantages that it serves as a warm-up for our scheme for the evolving k-THR access structure given in Section 5 and it is linear (whereas the scheme from Corollary 4.2 is non-linear).

Theorem 4.4. *There is a secret sharing scheme for the evolving* 2-THR *access structure and an ℓ-bit secret in which the share size of the t^{th} party is* $\log t + (\ell + 1)\log\log t + 4\ell + 1$.

4.1 The Connection to Prefix Codes

Here we prove Theorem 4.1, the tight connection between schemes for the evolving 2-threshold access structure and prefix codes. We also deduce Corollaries 4.2 and 4.3, upper and lower bounds (respectively) on share size in schemes for the evolving 2-threshold access structure.

Proof of the "only if" part of Theorem 4.1. Let $\Sigma\colon \mathbb{N} \to \{0,1\}^*$ be a prefix code for the integers. That is, for any $t_1, t_2 \in \mathbb{N}$ such that $t_1 \neq t_2$, it holds that $\Sigma(t_1)$ is not a prefix of $\Sigma(t_2)$. For $t \in \mathbb{N}$ denote by $\sigma(t)$ the length of the codeword $\Sigma(t)$.

The scheme. Let $s \in \{0,1\}$ be the secret to be shared. Let w be an infinite random binary string. The dealer generates the string as needed: at time $t \in \mathbb{N}$ the dealer holds the prefix of length $\sigma(t)$ of the string w, denoted by $w_{[\sigma(t)]}$ (for simplicity we assume that $\sigma(t)$ is monotonically increasing, but this is not necessary). The share of party t is a string u_t such that:

1. If $s = 0$, then $u_t = w_{[\sigma(t)]}$.
2. If $s = 1$, then $u_t = \Sigma(t) \oplus w_{[\sigma(t)]}$ (bit-wise XOR).

The share size of the t^{th} player in this scheme is $\sigma(t)$.

To reconstruct the secret, any two different parties t_1 and t_2, holding shares u_1 and u_2, respectively, where $|u_1| \leq |u_2|$, should check if u_1 is a prefix of u_2. If it is a prefix, then they output $s = 0$ and otherwise, they output $s = 1$.

Correctness and security. If $s = 0$, then since u_1 and u_2 are both prefixes of the same string w it holds that u_1 is a prefix of u_2. On the other hand, if $s = 1$ then $u_1 = \Sigma(t_1) \oplus w_{[\sigma(t_1)]}$ and $u_2 = \Sigma(t_2) \oplus w_{[\sigma(t_2)]}$, where $w_{[\sigma(t_1)]}$ is a prefix of $w_{[\sigma(t_2)]}$. Since Σ is a prefix code, $\Sigma(t_1)$ is *not* a prefix of $\Sigma(t_2)$, and thus u_1 is not a prefix of u_2.

Security follows since for both $s = 0$ and for $s = 1$ each single party t holds a single string u_t which is uniformly distributed over $\{0,1\}^{\sigma(t)}$. In case $s = 0$ this is true by construction, and in case $s = 1$ this is true since all the party sees is the codeword $\Sigma(t)$ XORed with $w_{[\sigma(t)]}$ which is uniform. ■

Proof of Corollary 4.2 (an instantiation via Elias's code). We describe the prefix code of Elias [Eli75] as a two step solution (for simplicity, in the description we ignore rounding issues). The first step will be a "basic scheme" and the second step will recursively use the basic scheme to improve the size of the encoding (this is very similar to the strategy we use in the proof of

Corollary 4.2). In the basic scheme, a number $t \in \mathbb{N}$ is encoded in two parts: the first part is $\log t$ zeros (this is just a unary representation of the length of t) and the second part is the binary representation of t. In total, the size of the encoding is $2\log t + 1$.

In the second step, we encode the first part of the basic scheme recursively using the basic scheme itself. Specifically, we encode the unary representation of the length of t using $2\log\log t + 1$ bits via the basic scheme. This results with an encoding of size $\log t + 2\log\log t + 2$ (one can recursively apply this transformation and get an even shorter encoding). ■

Proof of the "if" part of Theorem 4.1. Assume that we are given a secret sharing scheme for the evolving 2-threshold access structure and a 1-bit secret in which the share size of the t^{th} party is $\sigma(t)$. A property that the function $\sigma(\cdot)$ must satisfy is implicit in the lower bound of Kilian and Nisan [KN90] (that appears in [CCX13, Appendix A]) on the share size in 2-out-of-n secret sharing schemes. It says that the share sizes in such a scheme must satisfy *Kraft's inequality* (see below).

Claim 4.5 (Implicit in [KN90] and [CCX13, Appendix A])**.** For any $n \in \mathbb{N}$, in any secret sharing scheme for $(2,n)$-THR, it holds that $\sum_{t=1}^{n} \frac{1}{2^{\sigma(t)}} \leq 1$, where $\sigma(t)$ is the share size of the t^{th} party.

It is known that Kraft's inequality gives a *necessary and sufficient* condition for the existence of a prefix code for a given set of codeword lengths.

Claim 4.6 ([CT06, Theorem 5.2.1])**.** For any prefix code over the alphabet $\{0,1\}$, the codeword lengths $\sigma(1),\ldots,\sigma(n)$ must satisfy $\sum_{t=1}^{n} \frac{1}{2^{\sigma(t)}} \leq 1$. Conversely, given a set of codeword lengths that satisfy this inequality, there exists a prefix code with these word lengths.

The proof of the sufficient direction is constructive: given the collection of lengths of codewords it is possible to construct the code. Furthermore, we do not need to know the collection of lengths in advance, i.e. we can create the code on the fly, as long as the demand ($\sum_t \frac{1}{2^{\sigma(t)}}$, where $\sigma(t)$ is the size of the encoding of the number t) does not exceed 1 [CT06, Theorem 5.2.2].

Thus, any secret sharing scheme for the evolving 2-THR access structure in which the share size of the t^{th} party is $\sigma(t)$, implies the existence of a prefix code in which the length of the t^{th} codeword is $\sigma(t)$ (however, efficiency might not be preserved). ■

Proof of Corollary 4.3 (the lower bound). Assume (towards contradiction) that there is a secret sharing scheme for the evolving 2-THR access structure in which the share size of the t^{th} party is at most $\sigma(t) = \log t + \log\log t + \ldots + \log^{(\ell)} t + c$ for constants $c, \ell \in \mathbb{N}$. We can use this scheme to implement a standard secret sharing scheme for $(2,n)$-THR in which the share size of party $t \in [n]$ is $\sigma(t)$. By Claim 4.5, we get that

$$1 \geq \sum_{t=1}^{n} \frac{1}{2^{\sigma(t)}} \geq \frac{1}{2^c} \cdot \sum_{t=1}^{n} \frac{1}{t \cdot \log t \cdot \ldots \cdot \log^{(\ell-1)} t}.$$

But, as $n \to \infty$ the right hand side is at least $\frac{1}{2^c} \cdot \int_1^{\infty} \frac{1}{\prod_{i=0}^{\ell-1} \log^{(i)}(t)} dt = \log^{(\ell)} t$, which is strictly larger than 1 for large enough t. This is a contradiction which proves our claim. ■

4.2 A Direct Construction

Here we prove Theorem 4.4, a direct construction of a secret sharing scheme for the evolving 2-THR access structure. The construction is based on the following "recursive composition" lemma.

Lemma 4.7. *Assume that there exists a secret sharing scheme for the evolving* 2-THR *access structure and an ℓ-bit secret in which the share size of the t^{th} party is $\sigma(t)$. Then, there exists a secret sharing scheme for the evolving* 2-THR *access structure in which the share size of the t^{th} party is*

$$\max\{\ell, \log t\} + \sigma(\log t + 1).$$

Proof. Let Π be a construction of a secret sharing scheme for evolving 2-THR in which the share size of the t^{th} party when sharing an ℓ-bit secret is $\sigma(t)$. Let $s \in \{0,1\}^\ell$ be the secret to be shared. Each party, when it arrives, is assigned to a generation. The generations are growing in size: For $g = 0, 1, 2 \ldots$ the g^{th} generation begins when the (2^g)-th party arrives. Therefore, the size of the g^{th} generation, namely, the number of parties that are part of this generation, is $\text{SIZE}(g) = 2^g$ and party $t \in \mathbb{N}$ is part of generation $g = \lfloor \log t \rfloor$.

When a generation begins the dealer prepares shares for all parties that are part of that generation. Let us focus on the beginning of the g^{th} generation and describe the dealer's procedure:

1. Split s using a secret sharing scheme for $(2, \text{SIZE}(g))$-THR. Denote the resulting shares by $u_1^{(g)}, \ldots, u_{\text{SIZE}(g)}^{(g)}$.
2. Generate one share using the secret sharing scheme Π given the secret s and previous shares $\{v^{(i)}\}_{i \in \{0,\ldots,g-1\}}$. Denote the resulting share by $v^{(g)}$.
3. Set the secret share of the j^{th} party in the g^{th} generation (i.e. $j \in [\text{SIZE}(g)]$) to be

$$\left(u_j^{(g)}, v^{(g)}\right).$$

The output of the scheme is depicted in Figure 1.

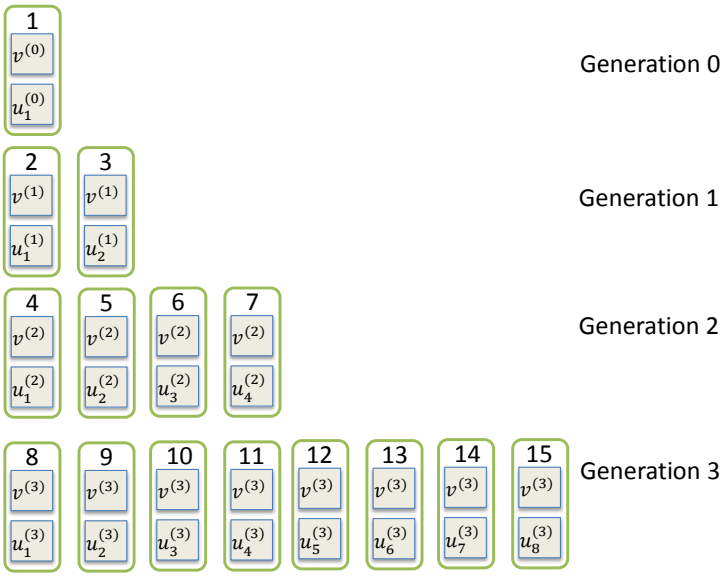

Figure 1: The shares of parties $1, \ldots, 15$ from generations $0, \ldots, 3$.

Correctness and security. Let $t_1, t_2 \in \mathbb{N}$ be any two different parties. If t_1 and t_2 are from the same generation g (i.e. if $g = \lfloor \log t_1 \rfloor = \lfloor \log t_2 \rfloor$), then they can reconstruct the secret s using the reconstruction procedure of the $(2, \text{SIZE}(g))$-THR scheme using the corresponding $u^{(g)}$ shares. If they are from different generations $g_1 \neq g_2$, then the parties can compute s using the reconstruction procedure of the evolving 2-THR scheme and the two shares $v^{(g_1)}$ and $v^{(g_2)}$.

For security consider any single party $t \in \mathbb{N}$ from generation g. We use Fact 2.3 and the assumptions that the schemes $(2, \text{SIZE}(g))$-THR and evolving 2-THR are secure, together with the fact that the sharing according to both of them is done independently, to get that the secret is perfectly hidden and conclude the proof.

Share size analysis. Fix party $t \in \mathbb{N}$ and assume that it is the j^{th} party of generation $g = \lfloor \log t \rfloor$. Its share is composed of two parts:

1. $u_j^{(g)}$ – generated by secret sharing s using a scheme for $(2, \text{SIZE}(g))$-THR. Since $\text{SIZE}(g) = 2^g$ and using Claim 2.4 we get that
$$|u_j^{(g)}| \leq \max\{\ell, \log(\text{SIZE}(g))\} \leq \max\{\ell, \log t\}.$$
2. $v^{(g)}$ – generated by generating one share of a secret sharing scheme Π for evolving 2-THR. Recall that g shares were generated for previous generations. Therefore,
$$|v^{(g)}| = \sigma(g+1) = \sigma(\lfloor \log t \rfloor + 1).$$

Thus, the total share size in the scheme Π' is bounded by $\max\{\ell, \log t\} + \sigma(\log t + 1)$. ∎

Proof of Theorem 4.4. We start with the scheme described in the end of Section 3 for sharing a 1-bit secret for the evolving 2-threshold access structure. We denote by $\Pi^{(0)}$ this scheme when used to share an ℓ-bit secret (by sharing each bit independently). The share size in this scheme is[5]
$$\sigma^{(0)}(t) = \ell t.$$

Using Lemma 4.7 we get a scheme $\Pi^{(1)}$ in which the share size of the t^{th} party is
$$\begin{aligned}\sigma^{(1)}(t) &= \max\{\ell, \log t\} + \sigma^{(0)}(\log t + 1) \\ &= \max\{\ell, \log t\} + \ell \log t + \ell \\ &\leq (\ell + 1) \log t + 2\ell.\end{aligned}$$

Applying Lemma 4.7 again we get a scheme $\Pi^{(2)}$ with shares of size
$$\begin{aligned}\sigma^{(2)}(t) &= \max\{\ell, \log t\} + \sigma^{(1)}(\log t + 1) \\ &\leq \max\{\ell, \log t\} + (\ell + 1) \log(\log t + 1) + 2\ell \\ &\leq \log t + (\ell + 1) \log \log t + 4\ell + 1.\end{aligned}$$

We note that by applying Lemma 4.7 more times one can push the multiplicative dependence on ℓ to even lower-order terms. ∎

[5] Alternatively, we can use the construction of [CT12] (see Section 1.3) which gives the same share size.

5 Evolving k-Threshold

In this section we give a construction of a secret sharing scheme for the evolving k-threshold access structure for general $k \in \mathbb{N}$. The scheme that we give is linear.

Theorem 5.1 (Theorem 1.2 restated)**.** *For every $k, \ell \in \mathbb{N}$, there is a secret sharing scheme for the evolving k-THR access structure and an ℓ-bit secret in which for every $t \in \mathbb{N}$ the share size of the t^{th} party is*

$$(k-1) \cdot \log t + 6k^4 \ell \log\log t \cdot \log\log\log t + 7k^4 \ell \log k.$$

Our approach follows the general blueprint of the one from Theorem 4.4. We start with a basic scheme that has good dependency on k but poor dependency on t, and use a domain reduction technique in order to obtain better dependency on t. The question is which basic scheme should we use. If we start with the scheme for the evolving k-threshold access structure in which the share size is exponential in t or k (which is what we get using the scheme from Theorem 3.1), then (after the recursive composition) the share size will depend *exponentially* on k.

To overcome this, in Section 5.1, we present a tailor-made construction for the evolving k-threshold access structure in which the share size of party t has *almost linear* dependence on $k \cdot t$. The recursive composition lemma appears in Section 5.2 and the proof of Theorem 5.1 appears in Section 5.3.

5.1 The Basic Scheme

The main result of this subsection is a construction of a secret sharing scheme for the evolving k-THR access structure in which the share size of party t is roughly linear in $k \cdot t$.

Lemma 5.2. *For every $k, \ell \in \mathbb{N}$, there is a secret sharing scheme for the evolving k-THR access structure and an ℓ-bit secret in which for every $t \in \mathbb{N}$ the share size of the t^{th} party is bounded by $kt \cdot \max\{\ell, \log(kt)\}$.*

Proof. Our scheme can be viewed as a variant of the scheme for general access structures from Section 3 with an additional mechanism of generations. Our main idea is not to consider all possible combinations of k parties (as in Section 3), but to group parties into generations, ignore the identities of the parties within a generation, and only focus on their *quantity*.

Each party, when it arrives, is assigned to a generation. Party $t \in \mathbb{N}$ is assigned to generation $g = \lfloor \log_k t \rfloor$. The generations are growing in size: For $g = 0, 1, 2 \ldots$ the g^{th} generation begins when the k^g-th party arrives. Therefore, the size of the g^{th} generation (i.e. the number of parties that are members of this generation), is $\text{SIZE}(g) = k^{g+1} - k^g = (k-1) \cdot k^g$.

Let $s \in \{0,1\}^\ell$ be the secret. When a generation g begins the dealer remembers k^g ℓ-bit strings s_A for all $A = (c_0, \ldots, c_{g-1}) \in \{0, \ldots, k\}^g$ (where if $g = 0$ it remembers only the secret). Intuitively, each such s_A is an ℓ-bit string that we share to the parties in generation g assuming that in generation $i \in \{0, \ldots, g-1\}$ c_i parties arrived. We explain how the dealer sets the value of s_A for $A = (c_0, \ldots, c_g)$. Notation: let $s_{\mathsf{prev}(A)} = s$ if $g = 0$ and $s_{\mathsf{prev}(A)} = s_{(c_0,\ldots,c_{g-1})}$ otherwise.

1. If $c_g = 0$, set $s_A = s_{\mathsf{prev}(A)}$ and HALT.
2. If $c_0 + \ldots + c_g < k$, then the dealer:

(a) samples $r_A \leftarrow \{0,1\}^\ell$ uniformly at random.

(b) sets $s_A = s_{\mathsf{prev}(A)} \oplus r_A$.

(c) shares the ℓ-bits r_A among the parties in the g^{th} generation using Shamir's $(c_g, \text{SIZE}(g))$-THR secret sharing scheme.

3. If $c_0 + \ldots + c_g = k$, then the dealer shares the ℓ-bit string $s_{\mathsf{prev}(A)}$ among the parties in the g^{th} generation using Shamir's $(c_g, \text{SIZE}(g))$-THR secret sharing scheme.

For correctness, consider any minimal qualified set of parties, namely a set of k parties. Let g be the generation of the latest party and let $A = (c_0, \ldots, c_g)$ be a tuple of numbers that indicates how many parties arrive from each generation. It holds that $\sum_{i=0}^{g} c_i = k$ and without loss of generality $c_g > 0$. We show that these parties can recover s, as required. Indeed, the c_g parties from generation g can recover $s_{\mathsf{prev}(A)}$ which is equal to $s \oplus r_{(c_0)} \oplus r_{(c_0,c_1)} \oplus \ldots \oplus r_{(c_0,\ldots,c_{g-1})}$ (or it is exactly s if $g = 0$ in which case we are done). Now, the c_0 parties from generation 0 can recover $r_{(c_0)}$, the c_1 parties from generation 1 can recover $r_{(c_0,c_1)}$, and so on. Thus, all present parties together can recover s, as required.

The proof of security is by induction, in the spirit of the proof of security of the scheme for general access structures given in Section 3. We assume (by induction) that the scheme is secure for g generations and all thresholds up to k and prove that the scheme is secure for $g+1$ generations and all thresholds up to k. The base case follows immediately from the security of Shamir's scheme. After the dealer generates shares for generation 0, it holds k ℓ-bit string: $s_{(0)}, s_{(1)}, \ldots, s_{(k-1)}$, where $s_{(0)} = s$ is the original secret and where for $i \in [k-1]$ $s_{(i)} = s \oplus r_{(i)}$ for a random $r_{(i)}$. Namely, each such $s_{(i)}$ is an independent one-time pad of the secret, where each pad is shared among the parties in the 0^{th} generation such that any i of them can recover it (using Shamir's scheme).

The remaining procedure of the dealer consists of independently sharing each such $s_{(i)}$ via the evolving $(k-i)$-THR access structure among g generations. Since the set controlled by the adversary is unqualified, for each such secret $s_{(i)}$, either the padded secret or the pad cannot be recovered, either by the security of Shamir's scheme or by the security of the scheme guaranteed by the induction hypothesis, respectively. Thus, since the sharing among different secrets is done independently, we conclude by Fact 2.3 that the secret is perfectly hidden given the whole set of shares seen by the adversary.

Share size analysis. The share of party t from generation g is composed of k^{g+1} shares generated via standard threshold schemes over $\text{SIZE}(g)$ parties. Thus, in total, the share size of party t is bounded by $k^{g+1} \cdot \max\{\ell, \log(\text{SIZE}(g))\}$. Recall that $g = \lfloor \log_k t \rfloor$ and $\text{SIZE}(g) = (k-1) \cdot k^g$. Therefore, the share size is bounded by

$$k \cdot t \cdot \max\{\ell, \log((k-1) \cdot t)\} \leq kt \cdot \max\{\ell, \log(kt)\}.$$

■

5.2 Recursive Composition

Lemma 5.3. *Fix $k, \ell \in \mathbb{N}$. Assume that there exists a secret sharing scheme for the evolving k-THR access structure and an ℓ-bit secret in which the share size of the t^{th} party is $\sigma(t)$. Then, there exists a secret sharing scheme for the evolving k-THR access structure and an ℓ-bit secret in which the share size of the t^{th} party is at most $(k-1) \cdot \log t + k \cdot \sigma(\log t + k) + k^2 + \ell$.*

Proof. Let Π be a secret sharing scheme for evolving k-THR in which the share size of the t^{th} party is $\sigma(t)$. Let $s \in \{0,1\}^\ell$ be the secret to be shared. Each party is assigned to a generation. The generations are growing in size: For $g = 0, 1, 2 \ldots$ the g^{th} generation begins when the $2^{(k-1)\cdot g}$-th party arrives. Thus, party $t \in \mathbb{N}$ is part of generation $g = \lfloor (\log t)/(k-1) \rfloor$, and the number of parties that are part of generation g, is

$$\text{SIZE}(g) = 2^{(k-1)\cdot(g+1)} - 2^{(k-1)\cdot g} = 2^{(k-1)\cdot g} \cdot (2^{k-1} - 1) \leq t \cdot 2^{k-1}.$$

When a generation begins the dealer prepares shares for all parties that are members of that generation. We focus on the beginning of generation g and describe the dealer's procedure:

1. Split s using a secret sharing scheme for $(k, \text{SIZE}(g))$-THR. Denote the resulting shares by $u_1^{(g)}, \ldots, u_{\text{SIZE}(g)}^{(g)}$.
2. Generate $k-1$ shares using the secret sharing scheme Π given the secret s and previous shares $\{v_j^{(i)}\}_{i \in [g-1], j \in [k-1]}$. Denote the resulting shares by $v_1^{(g)}, \ldots, v_{k-1}^{(g)}$.
3. For $i \in [k-1]$, split $v_i^{(g)}$ using a secret sharing scheme for $(i, \text{SIZE}(g))$-THR. Denote the resulting shares by $\{w_{i,1}^{(g)}, \ldots, w_{i,\text{SIZE}(g)}^{(g)}\}_{i \in [k-1]}$.
4. Set the secret share of the j^{th} party in the g^{th} generation (i.e. $j \in [\text{SIZE}(g)]$) to be

$$\left(u_j^{(g)}, w_{1,j}^{(g)}, \ldots, w_{k-1,j}^{(g)}\right).$$

Correctness and security. We show that any k parties can recover the secret. If all the parties come from the same generation g, then they can use their $u^{(g)}$ in order to run the reconstruction procedure of the $(k, \text{SIZE}(g))$-THR scheme and recover s.

For k parties that come from at least two generations we show that they can jointly recover k shares for the evolving k-THR scheme Π. By correctness of Π, using these shares they can reconstruct s. Indeed, assume that c_0 parties come from generation 0, c_1 come from generation 1 and so on, where there is some generation g where $\sum_{i=0}^{g} c_i = k$ and for all i it holds that $c_i \leq k-1$.

Claim 5.4. Any $c \in [\text{SIZE}(g)]$ parties from generation g can compute $v_c^{(g)}$.

Proof. The c parties hold c shares for $(1, \text{SIZE}(g))$-THR scheme that give $v_1^{(g)}$, c shares for the $(2, \text{SIZE}(g))$-THR scheme that give $v_2^{(g)}$ and so on. ■

Using this claim we get that the k parties can recover $\sum_{i=0}^{g} c_i = k$ shares of the evolving k-THR scheme, as required.

For security, consider any set of $k-1$ parties B, where c_i parties come from generation i and $\sum c_i = k-1$. First, the u shares of the $(k, \text{SIZE}(g))$-THR scheme held by parties in B are independent of the secret (since B consists of less than k parties). Thus, by Fact 2.3, it remain to show that the secret is independent of the shares held by B related to the evolving k-THR scheme Π. Consider the view of parties in B coming from generation g, denoted by $B|_g$: they hold shares $\{w_{1,j}^{(g)}, \ldots, w_{k-1,j}^{(g)}\}_{j \in B_g}$, where $w_{i,\cdot}^{(g)}$ is an i-out-of-SIZE(g) share of $v_i^{(g)}$ which is a share of the evolving k-THR scheme. By the security of Shamir's scheme and since c_g parties are present from

generation g, the values of $v_{c+1}^{(g)}, \ldots, v_{k-1}^{(g)}$ are independent of the secret given all the other shares of parties in B (i.e. are perfectly hidden). Now, since $\sum c_i < k$ the concatenation of the shares $v_{c_i}^{(i)}$ consists of less than k shares for the evolving k-THR scheme which implies that the whole set of shares is independent of the secret.

Share size analysis. We bound the size of each component in the share of party t in the resulting scheme. The share of party t that is the j^{th} party of generation $g = \lfloor (\log t)/(k-1) \rfloor$ is composed of $u_j^{(g)}$ and $w_{1,j}^{(g)}, \ldots, w_{k-1,j}^{(g)}$:

1. $u_j^{(g)}$ – generated by secret sharing s using a scheme for $(k, \text{SIZE}(g))$-THR. By Claim 2.4, it holds that

$$|u_j^{(g)}| \leq \max\{\ell, \log(\text{SIZE}(g))\} \leq \max\{\ell, \log t + (k-1)\}$$

2. $w_{i,j}^{(g)}$ – generated by secret sharing $v_i^{(g)}$ using a scheme for $(i, \text{SIZE}(g))$-THR. By Claim 2.4, for $1 < i \leq k-1$ it holds that

$$|w_{i,j}^{(g)}| \leq \max\{\log(\text{SIZE}(g)), |v_i^{(g)}|\} \leq \max\{\log t + (k-1), |v_i^{(g)}|\}$$

   and for $i = 1$ it holds that

$$|w_{1,j}^{(g)}| = |v_i^{(g)}|.$$

   - $v_i^{(g)}$ – generated by generating a share of a sharing scheme Π for evolving k-THR. Recall that $g \cdot (k-1) \leq \log t + (k-1)$ shares were generated for previous g generations. Therefore, for all $i \in [k-1]$

$$|v_i^{(g)}| \leq \sigma(\log t + i) \leq \sigma(\log t + (k-1)).$$

   Therefore, for $1 < i \leq k-1$

$$|w_{i,j}^{(g)}| \leq \max\left\{\log t + (k-1), \sigma(\log t + (k-1))\right\}$$

   and for $i = 1$

$$|w_{1,j}^{(g)}| \leq \sigma(\log t + (k-1)).$$

Thus, the total share size in our scheme is bounded by:

$$\begin{aligned}
&\max\{\ell, \log t + (k-1)\} + \sigma(\log t + (k-1)) + (k-2) \cdot \max\{\log t + (k-1), \sigma(\log t + (k-1))\} \\
&\leq \log t + k + \ell + \sigma(\log t + k) + (k-2)(\log t + k + \sigma(\log t + k)) \\
&\leq (k-1)\log t + k \cdot \sigma(\log t + k) + k^2 + \ell.
\end{aligned}$$

■

5.3 Proof of Theorem 5.1

We use the scheme for evolving k-THR constructed in Section 5.1 in which the share size of the t^{th} party is $\sigma^{(0)}(t) = kt \cdot \max\{\ell, \log(kt)\}$. Using Lemma 5.3, we have a scheme $\Pi^{(1)}$ for evolving k-THR in which the share size of the t^{th} party is:

$$\sigma^{(1)}(t) = (k-1) \cdot \log t + k \cdot \sigma^{(0)}(\log t + k) + k^2 + \ell.$$

We bound $\sigma^{(0)}(\log t + k)$ next.

Claim 5.5. $\sigma^{(0)}(\log t + k) \leq 4k \log t \cdot \log\log t + 4k^2 \cdot \log(2k) + 2k^2 \ell \log t$.

Proof. If $k > \log t$, then

$$\sigma^{(0)}(\log t + k) \leq \sigma^{(0)}(2k) \leq 2k^2 \cdot \max\{\ell, \log(2k^2)\} \leq 4k^2 \log(2k) + 2k^2\ell.$$

If $k \leq \log t$, then

$$\begin{aligned}
\sigma^{(0)}(\log t + k) &\leq \sigma^{(0)}(2 \log t) \\
&\leq k \cdot 2 \log t \cdot \max\{\ell, \log(k \cdot 2 \log t)\} \\
&\leq 2k \log t \cdot \log\log t + 2k \cdot \log t \cdot \log(2k) + 2k\ell \log t \\
&\leq 4k \log t \cdot \log\log t + 4k^2 \log(2k) + 2k^2 \ell \log t,
\end{aligned}$$

where the last inequality follows since $2k \log t \cdot \log(2k) \leq 2k \log t \cdot \log\log t + 4k^2 \log(2k)$ and $2k\ell \log t \leq 2k^2 \ell \log t$. Together, we get that

$$\sigma^{(0)}(\log t + k) \leq \max\{\sigma^{(0)}(2 \log t), \sigma^{(0)}(2k)\} \leq 4k \log t \cdot \log\log t + 4k^2 \log(2k) + 2k^2 \ell \log t.$$

■

Hence, we get that

$$\begin{aligned}
\sigma^{(1)}(t) &= (k-1) \cdot \log t + k \cdot \sigma^{(0)}(\log t + k) + k^2 + \ell \\
&\leq (k-1) \cdot \log t + 4k^2 \log t \cdot \log\log t + 4k^3 \log(2k) + 2k^3 \ell \log t + k^2 + \ell \\
&\leq 5k^2 \log t \cdot \log\log t + 2k^3 \ell \log t + 5k^3 \log k + \ell.
\end{aligned}$$

Using Lemma 5.3 again, we get a scheme $\Pi^{(2)}$ in which the share size of the t^{th} party is

$$\sigma^{(2)}(t) = (k-1) \cdot \log t + k \cdot \sigma^{(1)}(\log t + k) + k^2 + \ell.$$

We bound $\sigma^{(1)}(\log t + k)$ as follows:

$$\begin{aligned}
\sigma^{(1)}(\log t + k) \leq& \max\{\sigma^{(1)}(2 \log t), \sigma^{(1)}(2k)\} \\
\leq& 5k^2 \cdot \log(2 \log t) \cdot \log\log(2 \log t) + 2k^3 \ell \log(2 \log t) + \\
& 5k^2 \cdot \log(2k) \cdot \log\log(2k) + 2k^3 \ell \log(2k) + 5k^3 \log k + \ell \\
\leq& 6k^2 \cdot \log\log t \cdot \log\log\log t + 3k^3 \ell \log\log t + 6k^3 \ell \log k + \ell.
\end{aligned}$$

Thus, we get that

$$\begin{aligned}\sigma^{(2)}(t) &= (k-1)\cdot\log t + k\cdot\sigma^{(1)}(\log t + k) + k^2\\ &\leq (k-1)\cdot\log t + 6k^3\log\log t\cdot\log\log\log t + 3k^4\ell\log\log t + 6k^4\ell\log k + 2\ell k^2\\ &\leq (k-1)\cdot\log t + 6k^4\ell\log\log t\cdot\log\log\log t + 7k^4\ell\log k.\end{aligned}$$

Remark. One can iteratively apply Lemma 5.3 again and again to decrease the dependence on $\log\log t\cdot\log\log\log t$. However, the dependence on $\log t$ cannot be improved using this method.

6 Further Work and Open Problems

This work suggests several research directions. The most evident one is to investigate the necessity of the linear dependence on k in the most significant term in our scheme for the evolving k-THR access structure. In particular, are more algebraic-oriented constructions possible?

The fact that our construction for general access structures in Theorem 1.1 results with shares of exponential size should come as no surprise, as the best constructions known for standard secret sharing schemes for general access structures have shares of exponential size (in the number of parties). Proving that shares of exponential size are necessary to realize some *evolving* access structure is an open problem. This task may be easier than proving lower bounds for non-evolving secret sharing schemes, especially when the access structure is chosen adaptively (see the remark after Definition 2.7). We note that our construction also works in this setting.

There are several interesting access structures for which we do not have efficient constructions. For example, a very natural evolving access structure, we call *dynamic threshold*, is the one in which qualified subsets are the ones which form a *majority* of the present parties at *some* point in time. The only scheme that realized this access structure we are aware of stems from our construction for general access structures from Section 3 which results with very long shares.

When $k = 2$, we show a tight connection between evolving secret sharing and prefix codes (see Section 4.1). Is there a generalization of prefix codes that is related to the evolving k-THR access structure for $k > 2$?

Secret sharing has had many applications in cryptography and distributed computing. One of the most notable examples is *multi-party computation* (MPC). Can secret sharing for evolving access structures be useful for MPC?

We focused on schemes in which correctness and security are *perfect*. One can relax correctness to work with high probability and to allow small statistical error in security. Can these relaxations be used to obtain more interesting and efficient schemes? Another variant of secret sharing schemes is the *computational* one. In these schemes security is required only against computationally bounded adversaries. Efficient computational schemes for much richer classes of access structures are known: for example, Yao's scheme [Yao] for general monotone circuits (described in [VNS+03, JKK+17]), Krawczyk's scheme [Kra93] for sharing long secrets under the threshold access structure, and Komargodski et al.'s scheme [KNY17] for general monotone NP access structures. Is there a meaningful way to define computationally-secure secret sharing schemes for evolving access structures? Can this be used to obtain efficient schemes for more classes of evolving access structures? Cachin [Cac95] studied a similar question in a model in which there is a large public bulletin board.

Other natural variants of secret sharing can be adapted to the evolving setting. For example, verifiable, robust and visual secret sharing. We leave these as interesting directions for future exploration.

Follow-up work. Recently, Komargodski and Paskin-Cherniavsky [KP17] proposed a scheme for the dynamic threshold access structure in which the share size of the t^{th} party structure is $O(t^4 \log t)$ bits. They also showed how to generically translate any scheme for the evolving k-threshold access structure (for any $k \in \mathbb{N}$) into a scheme which is robust (i.e. the shared secret can be recovered even if some parties hand-in incorrect shares).

Acknowledgments

We thank the IEEE Transactions on Information Theory reviewers for their very useful comments and suggestions.

References

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Technion - Israel Institute of Technology, 1996. http://www.cs.bgu.ac.il/~beimel/Papers/thesis.ps.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology - 3rd International Workshop, IWCC*, volume 6639, pages 11–46, 2011.

[BGK16] Andrej Bogdanov, Siyao Guo, and Ilan Komargodski. Threshold secret sharing requires a linear size alphabet. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, pages 471–484, 2016.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–10, 1988.

[BI01] Amos Beimel and Yuval Ishai. On the power of nonlinear secrect-sharing. In *16th Annual IEEE Conference on Computational Complexity, CCC*, pages 188–202, 2001.

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *8th Annual International Cryptology Conference, CRYPTO*, pages 27–35, 1988.

[Bla79] George R. Blakley. Safeguarding cryptographic keys. *Proceedings of the AFIPS National Computer Conference*, 22:313–317, 1979.

[BR88] Josh Cohen Benaloh and Steven Rudich. Unpublished. Private communication from Steven Rudich to Moni Naor, 1988.

[BY76] Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching. *Inf. Process. Lett.*, 5(3):82–87, 1976.

[Cac95] Christian Cachin. On-line secret sharing. In *IMA Conference*, pages 190–198, 1995.

[CCX13] Ignacio Cascudo Pueyo, Ronald Cramer, and Chaoping Xing. Bounds on the threshold gap in secret sharing and its applications. *IEEE Transactions on Information Theory*, 59(9):5600–5612, 2013.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology - EUROCRYPT*, pages 316–334, 2000.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.

[CT12] László Csirmaz and Gábor Tardos. On-line secret sharing. *Designs, Codes and Cryptography*, 63(1):127–147, 2012.

[DPT10] Yevgeniy Dodis, Mihai Patrascu, and Mikkel Thorup. Changing base without losing space. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC*, pages 593–602, 2010.

[Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[ER78] Shimon Even and Michael Rodeh. Economical encoding of commas between strings. *Communications of the ACM*, 21(4):315–317, 1978.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology - CRYPTO*, pages 339–352, 1995.

[ISN93] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *Journal of Cryptology*, 6(1):15–20, 1993.

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In *Advances in Cryptology - CRYPTO*, pages 133–163, 2017.

[KN90] Joe Kilian and Noam Nisan. Unpublished. See [CCX13], 1990.

[KN08] Gillat Kol and Moni Naor. Games for exchanging information. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC*, pages 423–432, 2008.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM J. Discrete Math.*, 5(4):596–603, 1992.

[KNY16] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, pages 485–514, 2016.

[KNY17] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. *J. Cryptology*, 30(2):444–469, 2017.

[KP17] Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: Dynamic thresholds and robustness. In *Theory of Cryptography - 15th International Conference, TCC 2017*, pages 379–393, 2017.

[Kra93] Hugo Krawczyk. Secret sharing made short. In *13th Annual International Cryptology Conference, CRYPTO*, pages 136–146, 1993.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *8th Annual Structure in Complexity Theory Conference*, pages 102–111, 1993.

[MMM02] Tal Malkin, Daniele Micciancio, and Sara K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *Advances in Cryptology - EUROCRYPT*, pages 400–417, 2002.

[PSW13] Rasmus Pagh, Gil Segev, and Udi Wieder. How to approximate a set without knowing its size in advance. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 80–89, 2013.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[VNS+03] V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *4th International Conference on Cryptology in India, INDOCRYPT*, pages 162–176, 2003.

[Yao] Andrew C. Yao. Unpublished. Mentioned in [Bei11]. See also [VNS+03, JKK+17].

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il