How to Share a Secret, Infinitely

Ilan Komargodski*, Moni Naor*, and 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 where the size of the share each party gets is roughly $\log n$ 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 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}.

1 Introduction

640K ought to be enough for anybody

Misattributed to Bill Gates, 1981

Engineering scalable systems is a delicate business: important decisions have to be made regarding balancing scalability and efficiency when fixing system parameters (such as the representation size of a date, the number of clients the system can serve simultaneously, security parameters and more). This inherent tradeoff between scalability and efficiency has had devastating consequences. There are many Y2K [34] style horror stories such as losing contact with the NASA spacecraft "Deep Impact" when its internal clock overflowed, triggering an endless series of computer reboots [20], and the IPv4 address exhaustion problems caused by the limited allocation size for numeric Internet addresses [33]. Can we design scalable systems without suffering a great deal of efficiency

* Research supported in part by grants from the Israel Science Foundation grant no. 1255/12, 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.

costs? In this work we investigate methods that do not assume a fixed upper bound on the number of participants in the area of secret sharing.

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 [2]. 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 [31] and Blakley [8] 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 [22] 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 [5] gave a scheme with share size polynomial in the monotone *formula* complexity of the access structure. Karchmer and Wigderson [24] 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.

Theorem 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}.*

[1] But see the work of Csirmaz and Tardos [15] discussed below.

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.

Theorem 2 (Informal). *There is a secret sharing scheme for the evolving k-threshold access structure and a 1-bit secret in which the share size of the t^{th} party is $(k-1) \cdot \log t + \mathsf{poly}(k) \cdot o(\log t)$.*

For $k = 2$, we present a construction for the evolving 2-threshold access structure with slightly better low order terms. In this scheme the share size of the t^{th} party is $\log t + \log\log t + 2\log\log\log t + O(1)$.[2] To complement this construction, we prove a matching lower bound showing that our scheme is tight.

Theorem 3. *For any constant $c \in \mathbb{N}$, there is* no *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 at most $\log t + \log\log t + c$.*

Finally, 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 4. *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 $\sigma(t)$.*

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 [22,5,24]. 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 [31]. In this scheme, to share a 1-bit secret among n

[2] See Sections 4 and 5 for efficient generalizations that support larger domains of secrets.

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 [12] for the original proof of Kilian and Nisan [25], an improvement, and a discussion of the history; see also [9]).

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 $p(i)$. 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 Theorem 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 essentially optimal share size for the t^{th} party: the first two high order terms are $\log t + \log\log t$ (without hidden constant factors) and there is an additional lower order term of $2\log\log\log t + 6$. See the simplified scheme in Section 4.

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 [1, Section 4.1]. Most of the known schemes are linear (see [3] for an exception). Linear schemes are very useful for updating and manipulating secret shares (cf. proactive secret sharing [21]) and have many applications, most notably for secure multi-party computation [4,14]. Our schemes from Theorems 1 and 2 are linear (see Section 5.5 for details), whereas the scheme based on prefix codes from Theorem 4 is non-linear.

1.2 Related work

Most similar to our setting is the notion of *on-line* secret sharing of Csirmaz and Tardos [15]. Csirmaz and Tardos present a scheme for any access structure in which 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 this scheme is linear

in d. 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. Our Theorem 2 is an exponential improvement on the latter.

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 [17] or the online encoding of Dodis et al. [16], labeling nodes for testing adjacency in possibly infinite graphs [23], forward-secure signatures with an unbounded number of time periods [29], and data structures for approximate set membership (Bloom filters) for sets of unknown size [30].

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 $\mathcal{A} = \mathcal{A}_1, \mathcal{A}_2, \ldots$ be any evolving access structure with corresponding monotone characteristic functions $f_1, f_2, \ldots$, where $f_t \colon \{0,1\}^t \to \{0,1\}$. 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 $w_{(b_1,\ldots,b_{t-1},1)}$, where $b_1, \ldots, b_{t-1} \in \{0,1\}$. The $w_{(b_1,\ldots,b_{t-1},1)}$'s are generated as follows: if party t "completes" a minimal qualified set whose indicator vector is $(b_1, \ldots, b_{t-1}, 1)$, then the dealer gives party t the bit $w_{(b_1,\ldots,b_{t-1},1)} = w_{(b_1,\ldots,b_{t-1})} \oplus \ldots \oplus w_{(b_1)} \oplus s$ (where $w_{(b_1,\ldots,b_i,0)} = 0$), so XORing the appropriate shares will recover s. Otherwise, if $(b_1, \ldots, b_{t-1}, 1)$ is unqualified, then the dealer sets $w_{(b_1,\ldots,b_{t-1},1)} \leftarrow \{0,1\}$ to be a uniformly random bit. See Section 3 for the exact details.

Evolving 2-threshold access structure. The approach of [15] 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 5). 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 access structure. There are several ideas underlying the generalization of the 2-threshold scheme to work for any k. As before, we assign each party 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 $\ell \leq k-1$ parties from some generation come together, they can reconstruct $s_1, \ldots, s_\ell$ which are ℓ 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 5). 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 which has share size roughly 2^t (or roughly t^k with the optimization described above), then the resulting scheme will have a factor that depends *exponentially* on k. This makes the scheme impractical even for small values of k.

A formula for the future. 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 construct, at least intuitively, a Boolean monotone formula for k-threshold that counts to k.[3] For this counting to make sense in the evolving setting we notice that counting to k can be done by summing up the number of 1's so far with the number of 1's that will come in the future. Since we are counting to k, both of these numbers can be bounded by k, so we have to prepare only k possibilities for the unknown future. To make this construction efficient, we combine it with a generation-like mechanism. See Section 5 for the full details.

Prefix codes and evolving 2-threshold. There are several clues that point to a connection with prefix codes: the construction with the repeated domain reduction is reminiscent of the Elias code construction; the lower bound on schemes for the evolving 2-threshold access structure in Theorem 3 uses what is identical to a Kraft inequality, which is a characterization of prefix codes. We are able to formalize this tight relationship:

- 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 $\sigma(t)$. Using the best prefix code constructions we get a scheme in which the share size is the same as in our direct construction described above (but it is less efficient for sharing secrets longer than 1 bit). See Section 7 for the transformation.
- 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 sufficient condition of Kraft's inequality yields prefix codes.

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 $\mathcal{X}$

[3] Even though we can make our construction a monotone formula, our final construction is not phrased as a formula since we want to optimize share size. To exemplify this gap notice that the secret sharing scheme that results from the best formula for k-threshold on n parties has share size $\mathsf{poly}(\mathsf{k}) \cdot \log n$ [19,10], while the scheme of Shamir has size roughly $\log n$, independently of k.

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

Definition 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.

The well known scheme of Shamir [31] for the (k, n)-THR access structure (based on polynomial interpolation) satisfies the following.

Claim 5 ([31]). *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 $\ell = m$.*[4]

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

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

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.

To define evolving access structures we need to define a restriction.

Definition 3 (Restriction). *Let $\mathcal{A}$ be an access structure on n parties and let $0 < m < n$. We denote by $\mathcal{A}|_m$ the* restriction *of $\mathcal{A}$ to the first m parties. That is,*

$$\mathcal{A}|_m = \{X \in \mathcal{A} \mid \{m+1, \ldots, n\} \cap X = \emptyset\}.$$

Due to monotonicity of the access structure, we have the following claim.

Claim 6. *If $\mathcal{A}$ is an access structure on n parties, then $\mathcal{A}|_m$ is an access structure over m parties.*

Proof. By definition of $\mathcal{A}|_m$, it contains only parties from the set $\{1, \ldots, m\}$. Thus, to prove the claim it is enough to show that $\mathcal{A}|_m$ is a monotone set, namely, that if $B \in \mathcal{A}|_m$ then for any $B \subseteq C \subseteq \mathcal{P}_m$. Indeed, since $\mathcal{A}$ is an access structure, for $B \in \mathcal{A}|_m$ and $B \subseteq C \subseteq \mathcal{P}_m \subseteq \mathcal{P}_n$, we have that $B, C \in \mathcal{A}$. By definition of $\mathcal{A}|_m$, it holds that $C \in \mathcal{A}|_m$.

Definition 4 (Evolving access structure). *A (possibly infinite) sequence of access structures $\{\mathcal{A}_t\}_{t\in\mathbb{N}}$ is called* evolving *if the following conditions hold:*

1. *For every $t \in \mathbb{N}$, it holds that $\mathcal{A}_t$ is an access structure over t parties.*
2. *For every $t \in \mathbb{N}$, it holds that $\mathcal{A}_t|_{t-1}$ is equal to $\mathcal{A}_{t-1}$.*[5]

This definition naturally gives rise to an evolving variant of threshold access structures (see Definition 2). Here, we think of k as fixed, namely, independent of the number of parties.

Definition 5 (Evolving threshold access structure). *For every $k \in \mathbb{N}$, let evolving k-*THR *be the* evolving *threshold access structure which contains for any access structure in the sequence all subsets 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 6 (Secret sharing for evolving access structures). *Let $\mathcal{A} = \{\mathcal{A}_t\}_{t\in\mathbb{N}}$ be an evolving access structure. Let S be a domain of secrets, where $|S| \geq 2$. A secret sharing scheme for $\mathcal{A}$ and S consists of a pair of algorithms* (SHARE, RECON)*. The sharing procedure* SHARE *and the reconstruction procedure* RECON *satisfy the following requirements:*

[5] Recall the definition of a *restriction* from Definition 3.

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.*

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.[6]

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 the sequence of access structures $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_t\}$ is evolving. In this variant, the RECON procedure gets the access structure $\mathcal{A}_t$ 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.

[6] 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 [26] in which the share size does not have a fixed upper bound.

2.2 Warm-up: undirected s-t-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 [6] (see also [2, §3.2]) 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 $w_u \oplus w_v$.

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

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

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. We emphasize that our construction also works in the scenario in which the access structure is chosen adaptively; see remark after Definition 6. We focus on the case where the secret is a single bit.

Theorem 7 (Theorem 1 restated). *For every evolving access structure there is a secret sharing scheme where the share size of the t^{th} party is at most 2^{t-1}.*

The fact that our construction 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 a very interesting open problem.

Proof of Theorem 7. Let $\mathcal{A} = \{\mathcal{A}_t\}_{t\in\mathbb{N}}$ be an evolving access structure.[7] Let $\{f_t\}_{t\in\mathbb{N}}$ be the sequence of functions, where $f_i\colon \{0,1\}^i \to \{0,1\}$ is the (monotone) characteristic function of $\mathcal{A}_i$.

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 (before party t arrives) the dealer maintains a set of bits we denote by $w_{(b_1,\ldots,b_i)}$ for all $i \in [t-1]$ and $b_1,\ldots,b_i \in \{0,1\}$. These bits are defined iteratively. First, the dealer sets $w_{(1)} = s$ if $f_1(1) = 1$ and it is a uniformly random bit otherwise. Moreover, for every $i \geq 1$, the dealer sets $w_{(b_1,\ldots,b_{i-1},0)} = 0$. The rest of the bits are defined as follows.

1. If $f_t(b_1,\ldots,b_{t-1},1) = 1$ and $f_{t-1}(b_1,\ldots,b_{t-1}) = 0$, then the dealer sets
$$w_{(b_1,\ldots,b_{t-1},1)} = w_{(b_1,\ldots,b_{t-1})} \oplus \ldots \oplus w_{(b_1)} \oplus s.$$
2. If $f_t(b_1,\ldots,b_{t-1},1) = 1$ and $f_{t-1}(b_1,\ldots,b_{t-1}) = 1$, then the dealer sets
$$w_{(b_1,\ldots,b_{t-1},1)} = 0.$$
3. If $f_t(b_1,\ldots,b_{t-1},1) = 0$, then the dealer sets
$$w_{(b_1,\ldots,b_{t-1},1)} \leftarrow \{0,1\}$$
to be a uniformly random bit.

The share of party $t \in \mathbb{N}$ consists of 2^{t-1} bits $w_{(b_1,\ldots,b_{t-1},1)}$ for all $b_1,\ldots,b_{t-1} \in \{0,1\}$.

Correctness and security. We argue correctness and security at time $t \in \mathbb{N}$. Let $\boldsymbol{b} = (b_1,\ldots,b_t) \in \{0,1\}^t$ be an indicator vector of a minimal qualified set of parties at time t. For every $i \in [t-1]$ such that $b_i = 1$, party i holds the bit $w_{(b_1,\ldots,b_i)}$. Party t, by construction, holds the bit $w_{(b_1,\ldots,b_{t-1})} \oplus \cdots \oplus w_{(b_1)} \oplus s$, where $w_{(b_1,\ldots,b_i,0)} = 0$ for $0 \leq i \leq t-2$. Therefore, by XOR-ing all the shares, namely, computing
$$\bigoplus_{i=1}^{t} w_{(b_1,\ldots,b_i)},$$
the parties present can compute s.

For security it is instructive to give a simple example that illustrates how the scheme works and why it is secure. Consider the access structure at time $t = 4$ that consists of the following qualified sets $\{\{1,2\},\{1,3\},\{1,4\},\{2,4\}\}$ and we

[7] As mentioned, our construction actually works in the setting where $\mathcal{A}_t$ itself is chosen at time t (and it is not known at any time $t' < t$).

will argue security for the set $\{3,4\}$. Party 1 is unqualified so its share is $w_{(1)}$ is a uniformly random bit. Party 2 completes a qualified set with party 1 and so it's share consists of two bits $w_{(0,1)}, w_{(1,1)}$, where $w_{(0,1)}$ is a uniformly random bit and $w_{(1,1)} = w_{(1)} \oplus s$. Similarly, the share of party 3 consists of four bits $w_{(0,0,1)}, w_{(0,1,1)}, w_{(1,0,1)}, w_{(1,1,1)}$, where $w_{(0,0,1)}$ and $w_{(0,1,1)}$, uniformly random, $w_{(1,1,1)} = 0$ since $\{1,2\}$ is qualified as well, and $w_{(1,0,1)} = w_{(1)} \oplus s$. Finally, the share of party 4 consists of 8 bits most of which are either 0 or uniformly random, and the interesting ones are $w_{(1,0,0,1)} = w_{(1)} \oplus s$ and $w_{(0,1,0,1)} = w_{(0,1)} \oplus s$. Let us assume that the shares given to parties $\{1,2,3\}$ do not reveal s and show that the shares of party 4 do not reveal it as well. Indeed, all its uniformly random bits and the zero bits do not help, so we focus on $w_{(1,0,0,1)}$ and $w_{(0,1,0,1)}$. We observe that since parties 4 and 3 both complete party 1 to be a qualified set, they both have the same share $w_{(1,0,0,1)} = w_{(1,0,1)} = w_{(1)} \oplus s$, so we can ignore $w_{(1,0,0,1)}$ as well and be left with $w_{(0,1,0,1)} = w_{(0,1)} \oplus s$. Now, the point is that since party 3 does not complete party 2 to get a qualified set, the element $w_{(0,1)}$ completely masks the secret. More generally, the formal vector space generated by the share $w_{(0,1,0,1)}$ is linearly independent of all other shares.

We sketch security in the general case by induction. For $t = 1$ it is immediate and assume that the scheme is secure for $t-1$. For every $(b_1, \ldots, b_t) \in \{0,1\}^t$, party t receives a bit $w_{(b_1,\ldots,b_{t-1},1)}$ which is either a uniformly random bit or the bit $w_{(b_1,\ldots,b_{t-1})} \oplus \cdots \oplus w_{(b_1)} \oplus s$, depending on the value of $f_t(b_1, \ldots, b_{t-1}, 1)$. Let $\boldsymbol{b} = (b_1, \ldots, b_t) \in \{0,1\}^t$ be an indicator vector of an unqualified set of parties at time t. Assume that $b_t = 1$, as the other case follows immediately from the induction hypothesis. For every $w_{(b_1,\ldots,b_{t-1},1)}$, the uniformly random bits given to party t do not give an unqualified set any additional information about the secret as they are independent of everything else this set posses, so we can ignore them. Let us consider all the bit of the form $w_{(b'_1,\ldots,b'_i)} \oplus \cdots \oplus w_{(b'_1)} \oplus s$ held by parties in $\boldsymbol{b}$. If there are two parties i, j such that $b_i = b_j = 1$ that complete the same set, then they posses the *same* bit so we can ignore one of them. We are left with the case in which all parties complete different subsets. In this case, one can see that all the shares are linearly independent and thus the secret remains hidden. Security follows by the hypothesis.

Share size. The share size of party t is 2^{t-1} bits. ■

3.1 Efficiency improvements

In some cases, depending on the access structure, it is possible to reduce the share size by slightly optimizing the above scheme. The 0 bits that occur due to Item 2, do not have to be remembered as they can be inferred from the access structure.

At time t, the shares of party t will consists of:

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 our 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. is the last one).

This optimization is useful for access structures in which the number of unqualified sets is small. For example, for the evolving 2-THR access structure, the fact that there are only t unqualified sets, implies a scheme in which 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 An Efficient Scheme for Evolving 2-Threshold

We now describe the efficient construction for a secret sharing scheme for the evolving 2-THR access structure. Recall that evolving 2-THR is the sequence of access structures $(2,1)$-THR, $(2,2)$-THR, $(2,3)$-THR, ... which allow, at any point in time, for every pair of parties to learn the secret while disallowing singletons to learn anything about it. We first focus on the case where the secret is a single bit and discuss the more general case in Section 4.1.

Theorem 8. *There is a secret sharing scheme for the evolving* 2-THR *access structure in which the share size of the t^{th} party is bounded by*

$$\log t + \log\log t + 2\log\log\log t + 6.$$

Recall that in the classical setting of secret sharing, where an upper bound on the number of parties is known, there is a very efficient scheme for $(2,n)$-THR in which the share size of each party is roughly $\log n$ (see Claim 5). In Section 6 we show that in the evolving setting, for any $c \in \mathbb{N}$, a scheme in which the share size of the t^{th} party is $\log t + \log\log t + c$ cannot exist. Thus, up to an additive $\log\log\log t$ term, our scheme is optimal.

Our main technical claim used to prove Theorem 8 is given in the following lemma.

Lemma 1. *Assume that there exists a secret sharing scheme for the evolving* 2-THR *access structure 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*

$$\log t + \sigma(\log t + 1).$$

Proof of Theorem 8 assuming Lemma 1. Recall that in Section 3 we constructed a secret sharing scheme for any evolving access structure that results with shares of size 2^{t-1}. However, using the efficiency improvements described in Section 3.1,[8] we get a scheme in which the share size of the t^{th} party is

$$\sigma^{(0)}(t) = t.$$

[8] Alternatively, we can use the construction of [15] (see Section 1.3) which gives the t^{th} party a share of size t.

Using Lemma 1 this gives rise to a scheme $\Pi^{(1)}$ in which the share size of the t^{th} party is

$$\begin{aligned}\sigma^{(1)}(t) &= \log t + \sigma^{(0)}(\log t + 1) \\ &= 2\log t + 1.\end{aligned}$$

Applying Lemma 1 again we get a scheme $\Pi^{(2)}$ in which the share size of the t^{th} party is

$$\begin{aligned}\sigma^{(2)}(t) &= \log t + \sigma^{(1)}(\log t + 1) \\ &\leq \log t + 2\log(\log t + 1) + 1 \\ &\leq \log t + 2\log\log t + 3.\end{aligned}$$

Applying Lemma 1 one last time we get a scheme $\Pi^{(3)}$ in which the share size of the t^{th} party is

$$\begin{aligned}\sigma^{(3)}(t) &= \log t + \sigma^{(2)}(\log t + 1) \\ &\leq \log t + \log(\log t + 1) + 2\log\log(\log t + 1) + 3 \\ &\leq \log t + \log\log t + 2\log\log\log t + 6.\end{aligned}$$

This proves the theorem.

We note that this bound is tight according to the lower bound in Theorem 3 up to the low-order term $\log\log\log t$.

We note that by applying Lemma 1 i times we can improve the share size for large enough t. This will match the lower bound up to a low order term of $\log^{(i)}(t)$ (See Remark 1). We choose to stop after three applications of Lemma 1 due to aesthetic reasons (but see Section 7). ■

We are left to prove Lemma 1.

Proof of Lemma 1. Let Π be a construction of a secret sharing scheme for evolving 2-THR in which the share size of the t^{th} party is $\sigma(t)$. We construct a scheme Π' for the same access structure in which the share size is $\log t + \sigma(\log t + 1)$. We proceed with the description of the scheme.

Let $s \in \{0,1\}$ 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.

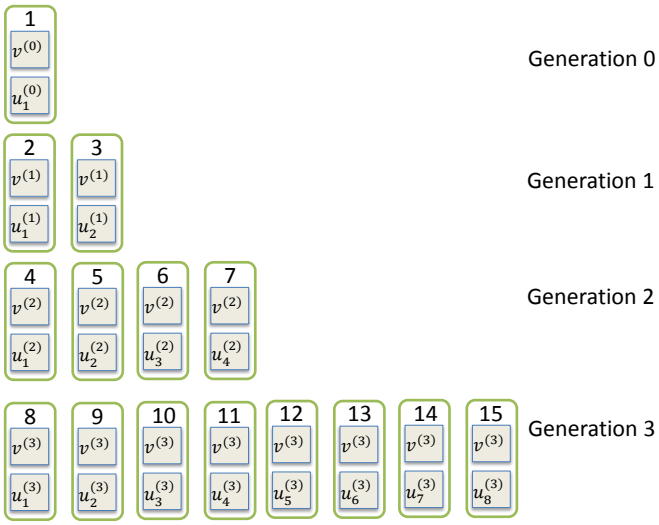

Fig. 1: The shares of parties $1, \ldots, 15$ from generations $0, \ldots, 3$.

Correctness and security. Let $t_1, t_2 \in \mathbb{N}$ be any two different parties. We show that the secret s can be computed from their shares. 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. By the security of the $(2, \text{SIZE}(g))$-THR scheme, the security of the evolving 2-THR scheme, and the fact that both parts of the share are generated independently, the shares cannot be used to learn anything about the secret.

Share size analysis. We analyze the share size of parties in the scheme Π'. Denote by $\sigma(t)$ the share size of party t in the scheme Π. We bound the size of each component in the share of party t. The share of party t that is the j^{th} party of generation $g = \lfloor \log t \rfloor$ is $(u_j^{(g)}, v^{(g)})$.

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 5 we get that

$$|u_j^{(g)}| \leq \log(\text{SIZE}(g)) \leq \lfloor \log t \rfloor.$$

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

$$\log t + \sigma(\log t + 1).$$

■

4.1 Generalization to larger domains of secrets

This scheme can be generalized to larger domains of secrets in an efficient way (in particular, better than sharing each bit independently). Roughly speaking, this follows since Shamir's threshold scheme can be used to share a secret longer than 1 bit without increasing the share size; see Claim 5. More generally, sharing a secret of ℓ-bits long, requires shares of size roughly $\max\{\log n, \ell\}$, where n is the number of parties in the scheme.

Let the secret be a string of length ℓ. Using the above feature of Shamir's scheme, a slight variant of Lemma 1 still holds (following the same proof). Namely, given any secret sharing scheme for the evolving 2-THR access structure and ℓ-bit secrets 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 and ℓ-bit secrets in which the share size of the t^{th} party is

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

We have to specify the initial scheme that supports ℓ-bit secrets to start the recursive composition with. We use our scheme for Theorem 8 by secret sharing every bit independently. The share size will be $\sigma^{(0)}(t) \leq \ell \cdot (\log t + \log\log t + 2\log\log\log t + 6)$. For large enough t it holds that $\sigma^{(0)}(t) \leq t$ and $\max\{\log t, \ell\} = \log t$. Thus, one can follow the same outline of the proof of Theorem 8 and obtain the *same* share size as in Theorem 8 for large enough t. (For smaller values of t one can follow the analysis and obtain a bound as a function of t and ℓ).

5 A Scheme for Evolving k-Threshold

In this section we give a construction for a secret sharing scheme for the evolving k-THR access structure for general k. As in Section 4, we first focus on the case where the secret is a single bit and discuss the more general case in Section 5.4.

Theorem 9 (Theorem 2 restated). *There is a secret sharing scheme for the evolving k-THR access structure in which the share size of the t^{th} party is at most*

$$(k-1)\cdot\log t + 6k^3\cdot\log\log t\cdot\log\log\log t + 7k^4\cdot\log k.$$

As in the case of $k=2$ (see the discussion after Theorem 8), the best one could hope to obtain is a scheme in which the share of the t^{th} party is close to $\log t$.[9] Our construction has a linear dependence on k and we leave open the question whether this can be improved.

We note that the bound in Theorem 9 applies for any $t\in\mathbb{N}$ and $k\geq 2$. For specific values of t and k it is possible to follow the analysis and obtain a better bound.

Our approach is to start with some basic scheme that has good dependency on k but high dependency on t and use a domain reduction technique in order to obtain better dependency on t.

Our main technical lemma used to prove Theorem 9 is a general transformation where we take any scheme for the evolving k-THR access structure (possibly with large share size), and convert it into a different scheme with smaller share size. Formally we prove following lemma.

Lemma 2. *Let $k\in\mathbb{N}$. Assume that there exists a secret sharing scheme for the evolving k-THR access structure 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 in which the share size of the t^{th} party is at most*

$$(k-1)\cdot\log t + k\cdot\sigma(\log t + k) + k^2.$$

The proof of Theorem 9 is done via repeated applications of Lemma 2, somewhat similarly to the proof of Theorem 8. However, naively the resulting parameters are not very good. Specifically, if we start with the scheme for the evolving k-THR access structure in which the share size is exponential in t or k (which is what we get using the scheme from Theorem 7; see Section 3.1), then by applying Lemma 2, the share size will eventually depend *exponentially* on k.

To overcome this, we first present a tailor-made construction for the evolving k-THR access structure in which the share size of party t has *almost linear* dependence on t and k. Using this scheme as a basic building block, we repeatedly apply Lemma 2 to obtain Theorem 9. The proof of the latter can be found in Section 5.3. The tailor-made construction for the evolving k-THR access structure appears next in Section 5.1. Finally, the proof of Lemma 2 appears in Section 5.2.

5.1 The basic scheme for evolving k-threshold

The main result of this subsection is a construction of a secret sharing scheme for the evolving k-THR access structure and 1-bit secrets in which the share size of party t is almost linear in t and k. This scheme will be used later as the basic building block in our final scheme for evolving k-THR satisfying Theorem 9.

[9] Shamir's scheme for (k,n)-THR results with shares of size roughly $\log n$. In particular, independent of k.

Lemma 3. *There is a secret sharing scheme for the evolving k-THR access structure in which the share size of the t^{th} party is bounded by $kt \cdot \log(kt)$.*

In the construction used to prove Lemma 3 we will employ two secret sharing schemes: (1) Shamir's threshold scheme and (2) a secret sharing scheme for a new access structure. The latter access structure C_ℓ over $2k$ parties, where $\ell \leq k$, is defined via its characteristic monotone function that we denote by C_ℓ as well. Let $(x, y) \in \{0,1\}^k \times \{0,1\}^k$ be an inputs to $\mathsf{C}_\ell \colon \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$, where we think of x and y as unary encoding of two numbers in $\{0, \ldots, k\}$. Jumping ahead, the variable x will represent the number of parties present so far and y will represent the number of parties to come. The access structure contains all pairs whose sum is at least ℓ. Formally, we define $\mathsf{C}_\ell(x, y) = 1$ if and only if at least one of the following conditions hold:

1. $\exists i, j \in [\ell - 1]$ such that $x_i = 1$, $y_j = 1$, and $i + j = \ell$.
2. $y_\ell = 1$ or $x_\ell = 1$.

Claim 10. *Let $\ell, k \in \mathbb{N}$ such that $\ell \leq k$. There exists a secret sharing scheme for the access structure C_ℓ in which the share size of each party is exactly the size of the shared secret.*

Proof. The following monotone formula computes C_ℓ:

$$\mathsf{C}_\ell(x, y) = \bigvee_{i=1}^{\ell-1} (x_i \wedge y_{\ell-i}) \vee (x_\ell \vee y_\ell) \,.$$

Notice that this formula is a DNF and every input variable appears exactly once. This formula gives rise to a simple secret sharing scheme for the access structure C_ℓ using the method of [5]. Since each variable appears at most once in the formula ($x_1, \ldots, x_\ell$ and $y_1, \ldots, y_\ell$ appear once, but $x_{\ell+1} \ldots, x_k$ and $y_{\ell+1}, \ldots, y_k$ do not appear), the share of each party is bounded by the length of the secret. The theorem follows by padding all shares to be of the same length.

Intuition for the construction. Our goal is to allow any combination of k parties to learn s. The main idea is not to consider all possible combinations of k parties, but to group parties into generations, ignore the identities of the parties within a generation, and only focus on their *quantity*. For simplicity, let us focus on the first and second generation. How many quantities should we consider? Exactly k, since the presence of $i \leq k$ parties from the first generation requires the presence of $k - i$ parties from the second generation. Therefore, the idea is to generate $2k$ strings $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$, such that only a proper combination of x_i and y_{k-i} will recover the secret s (for this we use the scheme for the access structure C_k). These $2k$ strings are generated when the first generation begins and the x's (the values corresponding to the "present") are shared among the parties of that generation in a way that allows any i parties to learn x_i. The y's (the values corresponding to the "future") will be shared

among the parties of the second generation in a similar way allowing any $k-i$ parties to learn y_{k-i}. Together, they will be able to recover s.

To formalize the above intuition and extend it to more generation we need some notation. For a generation $g \geq 0$, we denote by $[k]^g = \{1,\ldots,k\}^g$ the set $\underbrace{\{1,\ldots,k\} \times \ldots \times \{1,\ldots,k\}}_{g \text{ times}}$. We will be using vectors of the form $\boldsymbol{z} = (i_1,\ldots,i_g) \in [k]^g$ in our notation. For such a vector $\boldsymbol{z}$ and $i_{g+1} \in [k]$, we denote by $(\boldsymbol{z}, i_{g+1})$ the vector $(i_1,\ldots,i_g,i_{g+1}) \in [k]^{g+1}$.

Proof of Lemma 3. Let $s \in \{0,1\}$ be the secret to be shared. 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

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

When a generation g begins the dealer prepares shares for all parties that are members of that generation, and in addition, it generates k^{g+1} strings $\{y^{(g+1)}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in[k]^{g+1}}$ which it remembers for the next generation. Initially, the dealer sets $y^{(0)}_{\emptyset} = s$. Let us focus on the beginning of the g^{th} generation and describe the dealer's procedure (for consistency of notation we define $[k]^0 = \emptyset$):

1. (a) If $g = 0$: Split the string $y^{(0)}_{\emptyset} = s$ using the secret sharing scheme for C_k of Claim 10. Denote the resulting $2k$ shares by $x^{(0)}_{(1)},\ldots,x^{(0)}_{(k)},y^{(1)}_{(1)},\ldots,y^{(1)}_{(k)}$.
   (b) If $g \geq 1$: For all $\boldsymbol{z} = (i_1,\ldots,i_g) \in [k]^g$ split the string $y^{(g)}_{\boldsymbol{z}}$ using the secret sharing scheme for C_{i_g} of Claim 10. Denote the resulting $2k$ shares by $x^{(g)}_{(\boldsymbol{z},1)},\ldots,x^{(g)}_{(\boldsymbol{z},k)},y^{(g+1)}_{(\boldsymbol{z},1)},\ldots,y^{(g+1)}_{(\boldsymbol{z},k)}$.

   The x's will be shared amongst the parties in the current (g^{th}) generation, whereas the y's will be used to generate shares for parties in the next ($(g+1)^{\text{th}}$) generation.
2. For all $\boldsymbol{z} = (i_1,\ldots,i_{g+1}) \in [k]^{g+1}$ secret share $x^{(g)}_{\boldsymbol{z}}$ using a scheme for $(i_{g+1},\text{SIZE}(g))$-THR. Denote the resulting $\text{SIZE}(g)$ shares by $u^{(g)}_{\boldsymbol{z},1},\ldots,u^{(g)}_{\boldsymbol{z},\text{SIZE}(g)}$.
3. The secret share of the j^{th} party in the g^{th} generation (that is, the t^{th} party where $t = k^g + j - 1$) is composed of all the strings $u^{(g)}_{\boldsymbol{z},j}$ for any possible $\boldsymbol{z}$. Namely, it is the sequence of strings

$$\{u^{(g)}_{\boldsymbol{z},j}\}_{\boldsymbol{z}\in[k]^{g+1}}.$$

Correctness and security.

Claim 11. *Any $c \leq k$ parties from generation g can compute $\{x^{(g)}_{(\boldsymbol{z},i)}\}_{\boldsymbol{z}\in[k]^g, i\in[c]}$.*

Proof. Let $j_1, \ldots, j_c \in [\text{SIZE}(g)]$ be the indices of parties present from that generation. Thus, the parties can compute

$$\{u^{(g)}_{\boldsymbol{z},j_1}, \ldots, u^{(g)}_{\boldsymbol{z},j_c}\}_{\boldsymbol{z}\in[k]^{g+1}}.$$

Therefore, all the x values that were shared via a threshold scheme in which the threshold was at most c can be reconstructed. Namely, the values $\{x^{(g)}_{(\boldsymbol{z},i)}\}_{\boldsymbol{z}\in[k]^g, i\in[c]}$.

Claim 12. *Fix a generation $g \geq 0$, two numbers $c_1, c_2 \in [k]$ and $\boldsymbol{z} = (i_1, \ldots, i_g) \in [k]^g$. Then, given $x^{(g)}_{(\boldsymbol{z},c_1)}$ and $y^{(g+1)}_{(\boldsymbol{z},c_2)}$ such that $c_1 + c_2 \geq i_g$, one can compute $y^{(g)}_{\boldsymbol{z}}$. Moreover, given $x^{(g)}_{(\boldsymbol{z},c_1)}$ such that $c_1 \geq i_g$, one can compute $y^{(g)}_{\boldsymbol{z}}$.*

Proof. Follows from the correctness of the secret sharing scheme for C_ℓ.

Now, let us assume that k parties come together and try to reconstruct s. Assume that c_0 parties come from generation 0, c_1 come from generation 1 and so on. That is, for some generation g it holds that $\sum_{i=0}^{g} c_i = k$ and without loss of generality $c_g > 0$. We show that these parties can learn $y^{(0)}_{(k)} = s$, as required. This is done by applying Claims 11 and 12 iteratively. Details follow.

By Claim 11, using the shares of the c_i parties in generation $i \in \{0, \ldots, g\}$ we can compute

$$x^{(0)}_{(c_0)}, \quad \{x^{(1)}_{(\boldsymbol{z},c_1)}\}_{\boldsymbol{z}\in[k]^1}, \quad \ldots, \quad \{x^{(g)}_{(\boldsymbol{z},c_g)}\}_{\boldsymbol{z}\in[k]^g}.$$

By the second part of Claim 12, using $\{x^{(g)}_{(\boldsymbol{z},c_g)}\}_{\boldsymbol{z}\in[k]^g}$ we can reconstruct

$$\{y^{(g)}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in[k]^{g-1}\times\{c_g\}}.$$

By the first part of Claim 12, using $\{x^{(g-1)}_{(\boldsymbol{z},c_{g-1})}\}_{\boldsymbol{z}\in[k]^{g-1}}$ with $\{y^{(g)}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in[k]^{g-1}\times\{c_g\}}$ we can reconstruct

$$\{y^{(g-1)}_{\boldsymbol{z}}\}_{\boldsymbol{z}\in[k]^{g-2}\times\{c_g+c_{g-1}\}}.$$

Using the first part of Claim 12 iteratively as above, one can eventually compute $y^{(1)}_{(\sum_{i=1}^{g} c_i)}$. Combining with $x^{(0)}_{(c_0)}$, one can compute $y^{(0)}_{\emptyset} = s$, as required.

To argue security, fix any set of parties as above where $\sum_{i=0}^{g} c_i < k$. We claim that these parties cannot learn the value $y^{(0)}_{\emptyset} = s$. From the security of the scheme for C_ℓ, it is enough to show that they cannot learn any value in $y^{(1)}_{(k-c_0)}$. Applying this logic once again, it is enough to show that they cannot learn any value in $\{y^{(2)}_{(\boldsymbol{z},k-c_0-c_1)}\}_{\boldsymbol{z}\in[k]}$. Applying this argument g times, we get that s cannot be learned if and only if $\{y^{(g+1)}_{(\boldsymbol{z},k-\sum_{i=0}^{g} c_i)}\}_{\boldsymbol{z}\in[k]^g}$ cannot be learned. Indeed, these values are independent of the shares of parties up to generation g.

Share size analysis. We analyze the share size of parties in the scheme Π_k described above. 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 $k^{g+1} \cdot \log(\text{SIZE}(g))$. Recall that $g = \lfloor \log_k t \rfloor$ and $\text{SIZE}(g) = (k-1) \cdot k^g$. Therefore, the share size is bounded by

$$k \cdot t \cdot \log((k-1) \cdot t) \leq kt \cdot \log(kt).$$

■

5.2 Recursive composition: proof of Lemma 2

Let Π_k be a construction of a secret sharing scheme for evolving k-THR in which the share size of the t^{th} party is $\sigma_k(t)$. We construct a scheme Π'_k for the same access structure in which the share size is $\sigma'_k(t) = \log t + (k-1) + \sigma(\log t + (k-1)) + (k-2) \cdot \max\{\log t + (k-1), \sigma(\log t + (k-1))\}$.

Let $s \in \{0,1\}$ 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

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

As in Section 5.1, 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_k 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 learn 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 learn s. For k parties that come from at least two generations we show that they can jointly learn k shares for the evolving k-THR scheme Π_k. By correctness of Π_k, 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 13. *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 learn $\sum_{i=0}^{g} c_i = k$ shares of the evolving k-THR scheme, as required.

For security consider any set of $k-1$ parties. First, the u shares of the $(k, \text{SIZE}(g))$-THR scheme are independent of the secret. Thus, to complete the proof we need to show that the parties cannot learn any k shares of the evolving k-THR scheme Π_k. Indeed, any c parties from generation g cannot learn more than c shares $v_1^{(g)}, \ldots, v_c^{(g)}$; this follows from the security of the schemes $(c+1, \text{SIZE}(g))$-THR, $\ldots$, $(k-1, \text{SIZE}(g))$-THR. Therefore, in total, the parties can learn at most $\sum_{i=0}^{g} c_i < k$ shares.

Share size analysis. We bound the size of each component in the share of party t in the scheme Π'_k. 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 5 it holds that
$$|u_j^{(g)}| \leq \log(\text{SIZE}(g)) \leq \log t + (k-1)$$
2. $w_{i,j}^{(g)}$ – generated by secret sharing $v_i^{(g)}$ using a scheme for $(i, \text{SIZE}(g))$-THR. By Claim 5 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_k 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 $i \in [k-1]$
$$|v_i^{(g)}| \leq \sigma(\log t + (k-1)).$$
   Therefore, for $1 < i \leq k-1$
$$|w_{i,j}^{(g)}| \leq \max\{\log t + (k-1), \sigma(\log t + (k-1))\}$$
   and for $i = 1$
$$|w_{1,j}^{(g)}| \leq \sigma(\log t + (k-1)).$$

Thus, the total share size in the scheme Π'_k is bounded by:

$$\log t + (k-1) + \sigma(\log t + (k-1)) + (k-2) \cdot \max\{\log t + (k-1), \sigma(\log t + (k-1))\} \tag{1}$$
$$\leq \log t + (k-1) + \sigma(\log t + (k-1)) + (k-2)(\log t + (k-1) + \sigma(\log t + (k-1)))$$
$$\leq (k-1)\log t + k \cdot \sigma(\log t + k) + k^2.$$

5.3 Proof of Theorem 9 assuming Lemma 2

Let $k \in \mathbb{N}$ be such that $k \geq 2$. We use the scheme for evolving k-THR constructed in Section 5.1 in which the share size of the t^{th} party is $\sigma_k^{(0)}(t) = kt \cdot \log(kt)$. Using Lemma 2 this gives rise to a scheme $\Pi_k^{(1)}$ for evolving k-THR in which the share size of the t^{th} party is:

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

We bound $\sigma_k^{(0)}(\log t + k)$. If $k > \log t$, then

$$\sigma_k^{(0)}(\log t + k) \leq \sigma_k^{(0)}(2k) \leq 2k^2 \cdot \log\left(2k^2\right) \leq 4k^2 \cdot \log\left(2k\right)$$

If $k \leq \log t$ then

$$\begin{aligned}
\sigma_k^{(0)}(\log t + k) &\leq \sigma_k^{(0)}(2 \log t) \\
&\leq k \cdot 2 \log t \cdot \log\left(k \cdot 2 \log t\right) \\
&\leq 2k \cdot \log t \cdot \log\log t + 2k \cdot \log t \cdot \log\left(2k\right) \\
&\leq 4k \cdot \log t \cdot \log\log t + 4k^2 \cdot \log\left(2k\right),
\end{aligned}$$

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

$$\sigma_k^{(0)}(\log t + k) \leq \max\{\sigma_k^{(0)}(2 \log t), \sigma_k^{(0)}(2k)\} \leq 4k \cdot \log t \cdot \log\log t + 4k^2 \cdot \log\left(2k\right).$$

Plugging this in Equation (2), we get that

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

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

$$\sigma_k^{(2)}(t) = (k-1) \cdot \log t + k \cdot \sigma_k^{(1)}(\log t + k) + k^2. \tag{3}$$

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

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

Plugging this back in Equation (3), we get that

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

Remark. As in the proof of Theorem 8, one can iteratively apply Lemma 2 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.

5.4 Generalization to larger domains of secrets

Similarly to the generalization of the scheme from Section 4 to support larger domains of secrets (see Section 5.4), we generalize the above scheme. Let the secret be of length ℓ. Following the proof of Lemma 2, we obtain that given a scheme for the evolving k-THR access structure that supports secrets of length ℓ in which the share size of the t^{th} party is $\sigma(t)$, there exists a scheme for the same access structure and same length of secrets in which the share size of the t^{th} party is bounded by (cf. Equation (1))

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

Notice that for large enough $t \in \mathbb{N}$ the above bound is the *same* as the bound we had in Equation (1). For the recursive composition step (cf. Section 5.2) we start with the naive generalization of the scheme from Theorem 9 to support several input bits (i.e. bit by bit). This gives a scheme in which the share size is $\sigma^{(0)}(t) \leq \ell\cdot((k-1)\cdot\log t + k\cdot\sigma(\log t + k) + k^2)$. For large enough t it holds that $\sigma^{(0)}(t) \leq kt\cdot\log(kt)$. Thus, one can follow the same outline of the proof of Theorem 9 (see Section 5.3) and obtain the *same* share size as in Theorem 9 for large enough t. (For smaller values of t one can follow the analysis and obtain a bound as a function of t and ℓ).

5.5 Linearity of the scheme

The scheme from Theorem 7 is linear over $\mathsf{GF}(2)$. In the scheme from this section the shares are composed of several different parts each being an element coming from a different scheme. Consider the scheme from Section 5.1 (denoted by $\Pi_k^{(0)}$ in Section 5.3). Each share there is a composition of several linear schemes (the threshold scheme of Shamir and the scheme of Benaloh and Leichter). Since composition of linear schemes results with a linear scheme, the scheme is linear. Next, for the basic construction $\Pi_k^{(1)}$ in Section 5.3, each share is composed of several parts each being either a share of a linear scheme (Shamir's scheme) or a composition of linear schemes (Shamir's scheme and the scheme $\Pi_k^{(0)}$), resulting with a linear scheme. The same argument applies for the recursive composition which eventually gives that the final construction is linear.

6 A Lower Bound

For general access structures the best standard secret sharing schemes require exponential-size shares. Instantiating our scheme for n parties, results with the n^{th} party holding a share of size 2^{n-1}. Thus, any improvement in the share size on our scheme for general access structures, will imply a non-trivial improvement for general access structures in the standard setting.

In the case of k-threshold access structures, we do not know if our scheme is tight. Specifically, for $k > 2$, using our scheme to implement a standard secret sharing scheme for k-out-of-n is not tight. Indeed, the most significant term in the share size in our scheme depends linearly on $k - 1$, while the best schemes in the standard setting are independent of k (see Claim 5).

Thus, one may ask whether there exists a secret sharing scheme for the evolving k-THR in which the share size of the t^{th} party is roughly $\log t$. We show that such a scheme cannot exist.

Theorem 14 (Theorem 3 restated). *For any constant $c \in \mathbb{N}$, there is* no *secret sharing scheme for the evolving* 2-THR *access structure in which the share size of the t^{th} party is at most*

$$\log t + \log\log t + c.$$

Proof. 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 $\log t + \log\log t + c$ for a constant $c \in \mathbb{N}$. 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 $m_t \leq \log t + \log\log t + c$.

We use the following claim that underlies the lower bound of Kilian and Nisan. This inequality is the same as Kraft's (see [13, Chapter 5.2]), a fact that we use in Section 7.

Claim 15 ([25] and [12, Appendix A]). *For any $n \in \mathbb{N}$, in any secret sharing scheme for $(2, n)$-THR, it holds that*

$$\sum_{t=1}^{n} \frac{1}{2^{m_t}} \leq 1,$$

where m_t is the share size of the t^{th} party.

Using this claim we get that

$$1 \geq \sum_{t=1}^{n} \frac{1}{2^{m_t}} \geq \sum_{t=2}^{n} \frac{1}{2^{\log t + \log\log t + c}} = \frac{1}{2^c} \cdot \sum_{t=2}^{n} \frac{1}{t \cdot \log t}.$$

To get a contradiction we need to show that $\sum_{t=2}^{n} \frac{1}{t \cdot \log t} > 2^c$ for large enough n. Indeed, letting $n \to \infty$, we have that

$$\sum_{t=2}^{\infty} \frac{1}{t \cdot \log t} \geq \int_{2}^{\infty} \frac{1}{t \cdot \log t} dt = \log\log t \Big|_{2}^{\infty} \to \infty.$$

This completes the proof.

Remark 1 (A stronger lower bound). We note that the lower bound can be strengthened to show that even schemes in which the share size is $\sum_{t=1}^{\ell} \log^{(i)}(t) + c$ cannot exist for any $\ell \in \mathbb{N}$ and where $\log^{(i)}(t)$ is the i-times repeated log of t (letting $\log^{(0)}(t) = t$). This follows similarly to the above argument noting that for every $\ell \in \mathbb{N}_0$ it holds that $\int_1^{\infty} \frac{1}{\prod_{i=0}^{\ell} \log^{(i)}(t)} dt = \log^{(\ell+1)} t$ and using that $\log^{(\ell+1)} t \geq 2^c$ for any constant $c \in \mathbb{N}$ and large enough t.

This is reminiscent of bounds in the literature on prefix codes [7,18]. This is not surprising given the equivalence (in terms of complexity) between prefix codes and secret sharing for the evolving 2-THR access structures developed in Section 7.

7 The Equivalence Between Evolving 2-Threshold and Prefix Codes

We now show the very tight connection between schemes for the evolving 2-THR access structure and prefix codes.

Theorem 16 (Theorem 4 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 1-bit secret in which the share size of the t^{th} party is $\sigma(t)$.*

Proof of the "if" part of Theorem 16. Kraft's inequality (see [13, Theorem 5.2.2]) gives a *necessary and sufficient* condition for the existence of a prefix code for a given set of codeword 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 $(\sum_t \frac{1}{2^{m_t}})$ does not exceed 1. This inequality is the same as the one given in Claim 15 that must be satisfied by any secret sharing scheme for the evolving 2-THR access structure. 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 $\sigma(t)$. ■

Proof of the "only if" part of Theorem 16. 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).

Reconstruction. 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.

Share size. The share size of the t^{th} player in this scheme is $\sigma(t)$. Using the best constructions of prefix codes [7,18], we get the share size of Theorem 8.

Generalization to larger domains of secrets. One can support sharing of longer secrets by sharing every bit independently. Our direct construction presented in Section 4 is more efficient for sharing longer secrets (see Section 4.1 for more details). ■

Efficiency preservation. Note that the transformation from the prefix code to secret sharing preserves the efficiency of the code, i.e. dealing a share to party t is as easy as computing $\Sigma(t)$. However, the other direction, with the construction based on Kraft's inequality, does not preserve the efficiency. That is, we cannot say that encoding the number t, i.e. computing $\Sigma(t)$, is as easy as dealing a secret to party t.

8 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?

There are several interesting access structures for which we do not have efficient constructions. For example, a very natural evolving access structure 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 7). 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 [35,28,32,27]. 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 [11] 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.

References

1. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D. thesis, Technion - Israel Institute of Technology (1996), http://www.cs.bgu.ac.il/~beimel/Papers/thesis.ps
2. Beimel, A.: Secret-sharing schemes: A survey. In: Coding and Cryptology - 3rd International Workshop, IWCC. vol. 6639, pp. 11–46 (2011)
3. Beimel, A., Ishai, Y.: On the power of nonlinear secrect-sharing. In: 16th Annual IEEE Conference on Computational Complexity, CCC. pp. 188–202 (2001)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC. pp. 1–10 (1988)
5. Benaloh, J.C., Leichter, J.: Generalized secret sharing and monotone functions. In: 8th Annual International Cryptology Conference, CRYPTO. pp. 27–35 (1988)
6. Benaloh, J.C., Rudich, S.: Unpublished (1989), private Communication with Steven Rudich
7. Bentley, J.L., Yao, A.C.: An almost optimal algorithm for unbounded searching. Inf. Process. Lett. 5(3), 82–87 (1976)
8. Blakley, G.R.: Safeguarding cryptographic keys. Proceedings of the AFIPS National Computer Conference 22, 313–317 (1979)
9. Bogdanov, A., Guo, S., Komargodski, I.: Threshold secret sharing requires a linear size alphabet. Electronic Colloquium on Computational Complexity (ECCC) 131, 131 (2016), http://eccc.hpi-web.de/report/2016/131
10. Boppana, R.B.: Threshold functions and bounded depth monotone circuits. J. Comput. Syst. Sci. 32(2), 222–229 (1986)
11. Cachin, C.: On-line secret sharing. In: IMA Conference. pp. 190–198 (1995)
12. Cascudo Pueyo, I., Cramer, R., Xing, C.: Bounds on the threshold gap in secret sharing and its applications. IEEE Transactions on Information Theory 59(9), 5600–5612 (2013)
13. Cover, T.M., Thomas, J.A.: Elements of information theory (2. ed.). Wiley (2006)

14. Cramer, R., Damgård, I., Maurer, U.M.: General secure multi-party computation from any linear secret-sharing scheme. In: Advances in Cryptology - EUROCRYPT. pp. 316–334 (2000)
15. Csirmaz, L., Tardos, G.: On-line secret sharing. Designs, Codes and Cryptography 63(1), 127–147 (2012)
16. Dodis, Y., Patrascu, M., Thorup, M.: Changing base without losing space. In: STOC. pp. 593–602 (2010)
17. Elias, P.: Universal codeword sets and representations of the integers. IEEE Transactions on Information Theory 21(2), 194–203 (1975)
18. Even, S., Rodeh, M.: Economical encoding of commas between strings. Communications of the ACM 21(4), 315–317 (1978)
19. Friedman, J.: Constructing $O(n \log n)$ size monotone formulae for the k-th threshold function of n boolean variables. SIAM J. Comput. 15(3), 641–654 (1986)
20. Geographic, N.: NASA declares end to Deep Impact comet mission. http://news.nationalgeographic.com/news/2013/09/130920-deep-impact-ends-comet-mission-nasa-jpl/, accessed: 2016-02-07
21. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: Advances in Cryptology - CRYPTO. pp. 339–352 (1995)
22. Ito, M., Saito, A., Nishizeki, T.: Multiple assignment scheme for sharing secret. Journal of Cryptology 6(1), 15–20 (1993)
23. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. SIAM J. Discrete Math. 5(4), 596–603 (1992)
24. Karchmer, M., Wigderson, A.: On span programs. In: 8th Annual Structure in Complexity Theory Conference. pp. 102–111 (1993)
25. Kilian, J., Nisan, N.: Unpublished (1990), see [12]
26. Kol, G., Naor, M.: Games for exchanging information. In: STOC. pp. 423–432 (2008)
27. Komargodski, I., Naor, M., Yogev, E.: Secret-sharing for NP. In: Advances in Cryptology - ASIACRYPT 2014. pp. 254–273 (2014)
28. Krawczyk, H.: Secret sharing made short. In: 13th Annual International Cryptology Conference, CRYPTO. pp. 136–146 (1993)
29. Malkin, T., Micciancio, D., Miner, S.K.: Efficient generic forward-secure signatures with an unbounded number of time periods. In: Advances in Cryptology - EUROCRYPT. pp. 400–417 (2002)
30. Pagh, R., Segev, G., Wieder, U.: How to approximate a set without knowing its size in advance. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS. pp. 80–89 (2013)
31. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
32. Vinod, V., Narayanan, A., Srinathan, K., Rangan, C.P., Kim, K.: On the power of computational secret sharing. In: 4th International Conference on Cryptology in India, INDOCRYPT. pp. 162–176 (2003)
33. Wikipedia: IPv4 address exhaustion. https://en.wikipedia.org/wiki/IPv4_address_exhaustion, accessed: 2016-02-07
34. Wikipedia: Year 2000 problem. https://en.wikipedia.org/wiki/Year_2000_problem, acccessed: 2016-02-07
35. Yao, A.C.: Unpublished, mentioned in [2]. See also [32]