Evolving Secret Sharing: Dynamic Thresholds and Robustness

Ilan Komargodski[1(✉)] and Anat Paskin-Cherniavsky[2]

[1] Cornell Tech, New York, USA
komargodski@cornell.edu
[2] Department of Computer Science, Ariel University, Ariel, Israel
anatpc@ariel.ac.il

Abstract. Threshold secret sharing schemes enable a dealer to share a secret among n parties such that only subsets of parties of cardinality at least $k = k(n)$ can reconstruct the secret. Komargodski, Naor and Yogev (TCC 2016-B) proposed an efficient scheme for sharing a secret among an *unbounded* number of parties such that only subsets of k parties can recover the secret, where k is any fixed constant. This access structure is known as k-threshold. They left open the possibility of an efficient scheme for the *dynamic threshold* access structure, in which the qualified sets are of increasing size as the number of parties increases. We resolve this open problem and present a construction in which the share size of the t-th party is $O(t^4 \cdot \log t)$ bits.

Furthermore, we show how to generically translate any scheme for k-threshold into a scheme which is *robust*, where a shared secret can be recovered even if some parties hand-in incorrect shares. This answers another open problem of Komargodski et al. Our construction is based on the construction of robust (classical) secret sharing schemes of Cramer et al. (EUROCRYPT 2008) using algebraic manipulation detection codes.

1 Introduction

Secret sharing schemes, introduced by Shamir [17] and Blakley [5], are methods that enable a dealer, that holds a secret piece of information, to distribute this secret among n parties such that predefined qualified subsets can reconstruct the secret, while others learn nothing about it. The monotone collection of qualified subsets is known as an *access structure*. Secret sharing schemes are a basic primitive and have found numerous applications in cryptography and distributed computing; see the extensive survey of Beimel [2] and the book of Cramer et al. [9]. Any access structure admits a secret sharing scheme but the share size could be as large as $O(2^n)$, the maximal number of possible qualified sets [12].

This paper incorporates the manuscript of Paskin-Cherniavsky [15].

I. Komargodski—Supported in part by Elaine Shi's Packard Foundation Fellowship. Part of this work done while being a Ph.D student at the Weizmann Institute of Science, supported in part by grants from the Israel Science Foundation and by a Levzion Fellowship.

© International Association for Cryptologic Research 2017
Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part II, LNCS 10678, pp. 379–393, 2017.
https://doi.org/10.1007/978-3-319-70503-3_12

A significant goal in secret sharing is thus to minimize the share size, namely, the amount of information distributed to the parties.[1]

Almost all known secret sharing schemes assume that the number of parties n and the access structure are known in advance. However, in many scenarios these assumptions have a cost: First, the eventual set might turn out to be much smaller than n. Second, the access structure may change with time, forcing the dealer to re-share its secret. In a recent work, Komargodski et al. [14] initiated the study of secret sharing schemes for the case where the set of parties is *not* known in advanced and could potentially be infinite (or even more generally the access structure may change). Specifically, parties arrive one by one and whenever a 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. In the most general case, a qualified subset is revealed to the dealer only when the last party in that subset arrives. In special cases, the dealer knows the access structure to begin with, just does not have an upper bound on the number of parties. We assume that the changes to the access structure are monotone, namely, parties are only added and qualified sets remain qualified as more and more parties join. We call this an evolving access structure.

When designing a secret sharing scheme for an evolving access structure, the goal is to minimize the share size of the t^{th} party arriving as a function of t. Komargodski et al. showed that *any* evolving access structure can be realized albeit the share size of the t^{th} party is 2^{t-1}. Then, they consider 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 and showed an efficient scheme for it in which the share size of the t^{th} party is bounded by roughly $k \cdot \log t$ bits (see Theorem 2.5 for a precise statement). Their scheme was shown to be optimal in terms of share size for $k = 2$.

One of the main open problems left open by their work was to construct an efficient secret sharing scheme for the *evolving majority* access structure in which qualified subsets are the ones which form a *majority* of the present parties at *some* point in time. More precisely, a set of k parties with indices $i_1 < \ldots < i_k$ is qualified if and only if there exists an index $j \in [k]$ such that

$$|\{i_1, \ldots, i_j\}| \geq \frac{1}{2} \cdot i_j.$$

The $1/2$ threshold above is arbitrary and could be replaced with any other constant in $(0, 1)$ or even with a sequence of growing threshold $k_1 \leq k_2 \leq \ldots$ such that the qualified sets at time t are those sets of cardinality at least k_t. We resolve this open problem and construct a secret sharing scheme for this evolving majority access structure in which the share size of the t^{th} party is $O(t^4 \cdot \log t)$ bits. Our scheme is linear in the sense that reconstruction is done by applying a linear function on the shares [1, Sect. 4.1]. This property is desirable since it is useful in applications such as secure multiparty computation [3,8].

[1] Whether having exponentially large shares is necessary is a major open problem. The best lower bound known to date is (almost) linear by Csirmaz [11].

Another question left open in [14] was to construct *robust* secret sharing schemes for evolving access structures. In the setting described so far, secret sharing schemes assume the parties are honest and upon reconstruction provide their *correct* shares. However, in most cryptographic settings it is often the case that we need to handle malicious parties that manipulate their shares. For this, the strengthened notion of *robust* secret sharing was proposed by Ben-Or and Rabin [16]. This notion requires that the shared secret can be recovered even if some parties hand-in incorrect shares.

In the original construction of Ben-Or and Rabin each party authenticates the share of every other party using a MAC having unforgeability security $2^{-\lambda}$ (the reconstruction procedure checks that the majority of the tags are verified). When the number of parties is unbounded, it is unclear how to implement such a solution as the first party has to authenticate all future parties (which is an unbounded number). Several follow-up constructions of robust secret sharing schemes with smaller shares [4,6], rely on the same high-level idea of parties authenticating share of other parties (in a pairwise manner) and thus seem unsuitable for our setting.

We observe that a different line of works on robust secret sharing, ones based on algebraic manipulation detection (AMD) codes [7,10] can be adapted to the evolving setting. We thus present an efficient robust secret sharing scheme for the evolving k-threshold access structure such that as long as an adversary corrupts at most $k-1$ parties, from any set of $2k-1$ parties, one can recover the secret. The failure probability of our reconstruction procedure is $2^{-\lambda}$ and the share size is bounded by roughly $k \cdot \log t + \lambda$ bits.

2 Preliminaries

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 $\log 0 = -\infty$. For a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. A function $\mathsf{neg}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{neg}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$.

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 subsets. Subsets in* $\mathcal{A}$ *are called* qualified *and subsets not in* $\mathcal{A}$ *are called* unqualified.

A 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.

Definition 2.2. *A secret sharing scheme $\mathcal{S}$ for an access structure $\mathcal{A}$ consists of a pair of algorithms* $(\mathsf{SHARE}, \mathsf{RECON})$. SHARE *is a probabilistic procedure that gets as input a secret s (from a domain of secrets S such that $|S| \geq 2$) and a number n, and generates n shares $\Pi_1^{(s)}, \ldots, \Pi_n^{(s)}$.* RECON *is a deterministic procedure that gets as input the shares of a subset B and outputs a string. The requirements are:*

1. ***Correctness:*** *For every secret $s \in S$ and every qualified set $B \in \mathcal{A}$, it holds that*

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

   where the probability is over the randomness of the sharing procedure.
2. ***Security:*** *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 $\mathcal{S}$, denoted by $\mathsf{SS}(\mathcal{S})$, is the maximum number of bits each party holds in the worst case over all parties and all secrets. For an access structure $\mathcal{A}$ we denote by $\mathsf{SS}(\mathcal{A})$ the minimum of $\mathsf{SS}(\mathcal{S})$ over all schemes $\mathcal{S}$ for the access structure $\mathcal{A}$.

Linear schemes. An important subclass of secret sharing schemes are *linear* schemes. In such a 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, Sect. 4.1]. We denote by $\mathsf{lin\text{-}SS}(\mathcal{A})$ the minimum value of $\mathsf{SS}(\mathcal{S})$ over all *linear* schemes $\mathcal{S}$ for the access structure $\mathcal{A}$.

2.1 Evolving Secret Sharing

We recall the notion of an evolving access structure and the corresponding notion of secret sharing defined by [14]. Roughly speaking, these definitions capture the scenario in which the access structure is *not* fully known to the sharing procedure at once but is rather revealed in an online manner. Concretely, 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 2.3 (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 (as in Definition 2.1).*

Below we give a generalization of the definition of a standard secret sharing scheme (see Definition 2.2) to apply for evolving access structures as in [14]. 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 2.4 (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 $\mathcal{S}$ for $\mathcal{A}$ and S consists of a pair of algorithms $(\mathsf{SHARE}, \mathsf{RECON})$. The probabilistic sharing procedure SHARE and the deterministic 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.
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*

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

   where the probability is over the randomness of the sharing procedure.
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.[2]

In [14] it was shown how to construct a secret sharing scheme for any evolving access structure. This scheme results, for party t, with a share of size exponential in t. They further showed that in many special cases one can do much better. For example, in the evolving k-threshold access structure which contains all subsets of size k (where k is known), they gave a scheme in which the share size depends logarithmically on t.

[2] 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 [13] in which the share size does not have a fixed upper bound.

Theorem 2.5 [14]. *There is a secret sharing scheme for sharing a 1-bit secret for any evolving access structure in which for every $t \in \mathbb{N}$ the share size of the t^{th} party is 2^{t-1}.*

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

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 SHARE and RECON procedures get the access structure $\mathcal{A}_t$ as an additional parameter. An illustrative example where $\mathcal{A}_t$ is known ahead of time is the evolving k-threshold access structure mentioned above. (In this case k is fixed and is independent of t.) We will consider (in Sect. 3) a natural generalization in which there is a sequence of growing thresholds $k_1 < k_2 \ldots$ that say how many parties should be present as a function of the indices of the present parties themselves. This sequence of thresholds does not have to be known in advance.

2.2 Algebraic Manipulation Detection Codes

In our robust evolving secret sharing scheme we will use algebraic manipulation codes [10]. Originally, they were used to transform standard secret sharing schemes into robust ones.

Definition 2.6. *An (S, G, δ)-AMD code is a probabilistic encoding map $E\colon \mathcal{S} \to \mathcal{G}$ for a set $\mathcal{S}$ of size S and a group $\mathcal{G}$ of size G together with a deterministic decoding function $D\colon \mathbb{Z}_G \to [S] \cup \{\perp\}$ such that $D(E(s)) = s$ with probability 1 for every $s \in [S]$. Furthermore, for any $s \in [S]$ and $\Delta \in \mathbb{Z}_G$ it holds that*

$$\Pr_E[D(E(s) + \Delta) \notin \{s, \perp\}] \leq \delta.$$

The AMD code is called systematic *if $\mathcal{S}$ is a group, the encoding is of the form $E\colon \mathcal{S} \to \mathcal{S} \times \mathcal{G}_1 \times \mathcal{G}_2$ and $E(s)$ has the form $(s, x, f(x, s))$ for some function f and $x \in_R \mathcal{G}_1$. The decoding function of a systematic AMD code is given by $D(s', x', \sigma') = s'$ if $\sigma' = f(s', x')$ and $\perp$ otherwise.*

Theorem 2.7 [10]. *Let $\mathbb{F}$ be a field of size q and characteristic p, and let d be an integer such that $d + 2$ is not divisible by p. There exists a construction of a systematic $(q^d, q^{d+2}, (d+1)/q)$-AMD code. The encoding function maps $\mathbb{F}^d$ to $\mathbb{F}^d \times \mathbb{F} \times \mathbb{F}$.*

To achieve error parameter γ, and input domain S we will instantiate the above scheme with $\mathcal{G} = \mathbb{F}_2^t, d = 1$ where $t = \log S + \gamma + O(1)$. We refer to this construction as $\mathsf{AMD}_{S,\gamma}$.

3 A Scheme for Dynamic Threshold

In this section we present a secret sharing scheme for the evolving dynamic threshold access structure. This access structure is parametrized by a sequence of threshold values $k_1 \leq k_2 \leq \ldots$ such that at time t the qualified sets are those of cardinality at least k_t. The condition that $k_t \leq k_{t+1}$ is necessary for the monotonicity of the sequence of access structures, namely for the sequence of access structures to be a valid evolving structure.

Definition 3.1 (Dynamic threshold). *The dynamic threshold access structure is parametrized by a (possibly infinite) sequence of number $k_1 \leq k_2 \leq \ldots$. For any $t \in \mathbb{N}$, the set $\mathcal{A}_t$ of qualified sets at time t contains all those sets of cardinality at least k_t.*

Of particular interest is the following special case of dynamic threshold access structures in which the threshold at any point in time is a fixed function. Specifically, the function that we focus on is the one in which in time t the qualified sets are those of cardinality at least $\gamma \cdot t$ for fixed $\gamma \in (0, 1)$.

Definition 3.2 (γ-dynamic threshold). *For a parameter $\gamma \in (0, 1)$, the γ-dynamic threshold access structure is the above dynamic threshold access structures with sequence of numbers $\gamma \cdot 1, \gamma \cdot 2, \ldots$. That is, k parties $i_1 < \cdots < i_k$ is qualified iff there exists an index $j \in [k]$ such that $|\{i_1, \ldots, i_j\}| \geq \gamma \cdot i_j$.*

The main result of this section is summarized in the following theorem:

Theorem 3.3. *For any sequence of threshold values $\{k_t\}_{t\in\mathbb{N}}$ that define a dynamic threshold access structures, there exists a secret sharing scheme for sharing a 1-bit secret in which the share size of the t-th party is bounded by $O(t^4 \cdot \log t)$ bits.*

High level idea. The main idea is to represent the access structure as an infinite decision tree where the nodes in layer i are labeled by x_i. Turning such an infinite decision tree into an evolving secret sharing scheme can be done essentially generically via an evolving secret sharing scheme for undirected st-connectivity. This was done somewhat implicitly in [14] so we omit details here, but we just mention that the eventual share size is proportional to the tree size. Thus, using this naively gives us not very efficient schemes. In particular, for the dynamic threshold scheme it gives a scheme with exponential share size.

To improve this we observe that this decision tree can be "squashed" such that now each layer is labeled by a sequence of variables $x_i, \ldots, x_j$ and not just x_i. We call such a sequence a *generation*. Now, since every layer is labeled by a sequence of variables, we define each edge to be some *monotone* Boolean function of the variables in the generation. This operation potentially reduces the number of edges in the tree. If, in addition, this monotone function is simple enough (i.e. there is an efficient secret sharing scheme for it), this will eventually reduce the share size of our construction. Indeed, we can share the secret according to the new decision tree (with the squashed layers) to a virtual set of (much fewer)

parties that correspond to the squashed sets and then re-sharing those shares via a secret sharing scheme among the parties inside a generation.

In the case of dynamic majority, each edge between two generation is labeled by the number of parties in the generation that arrived. This is the only information we need to remember for each generation in our structure. Now, if enough parties come so that we can reconstruct the secret, the decision tree must contain a path that leads to an accepting node (and vice versa). Luckily, this access structure (that counts how many parties arrived from a specific generation) can be implemented very efficiently using Shamir's scheme.

It remains to explain how we set the size of a generation. If we set it too low, then we do not save much in the decision tree size. If we set it too high, then we have a lot of parties in each generation and the first party in that generation will have to pay too much. The exact choice really depends on the access structure in hand, but it turns out that for the dynamic threshold case, the optimal setting of generation size is so that it increases in a specific polynomial rate, namely, the i-th generation size is square of the $(i-1)$-th generation size.

The above overview was slightly over-simplified and the actual construction requires some more care. In particular, we present the scheme directly and not as a composition of many schemes as it does not require familiarity with the st-connectivity scheme, and it allows us to prove its security directly via induction.

Proof. We begin by recalling Shamir's scheme [17] which will be heavily used in our scheme. Shamir's scheme is a scheme for sharing a 1-bit secret s among n parties for the k-out-of-n access structure (which contains all subsets of cardinality at least k). The share size in his scheme is $\log q$ bits, where $q > n$ is a prime number (or a power of a prime). We denote this scheme by $\mathsf{Shamir}(n, k, s)$. Note that in the cases where $k = 1$ or $k = n$, there are more efficient schemes: for $k = 1$, each party gets the secret and for $k = n$, each party gets a random value conditioned on their XOR being the secret. In these cases, the share size is a single bit (and it is, in particular, independent of n).

We assign to each arriving party $t \in \mathbb{N}$ a generation $\mathsf{GenOf}(t)$. The size of generation i is doubly exponential, namely, $\mathsf{GenSz}(i) = 2^{2^i}$. Thus, the t-th party is part of the $\lceil \log \log t \rceil$-th generation (at most) which includes at most t^2 parties. The first party in generation g is $\sum_{i=1}^{g} \mathsf{GenSz}(i) = \sum_{i=1}^{g} 2^{2^i}$. The state of the dealer after generation g ends consists of strings s_A, where A ranges over all tuples $(c_0, \ldots, c_g)$ such that $c_i \in [2^{2^i}]$. In other words, the dealer maintains a string s_A for each $A = (c_0, \ldots, c_g) \in [\mathsf{GenSz}(0)] \times \ldots \times [\mathsf{GenSz}(g)]$, where $\mathsf{GenSz}(i) = 2^{2^i}$. The number c_i, in some sense, represents the number of parties present from generation i.

For the i^{th} party in the g^{th} generation, denote by $\mathsf{IdxOf}(G, i)$ the overall index of this party since the beginning of time. Denote by s the secret to be shared and set $s_{(0)} = s$. When the $(g+1)$-th generation begins, the dealer does the following for every $(c_0, \ldots, c_g) \in [\mathsf{GenSz}(0)] \times \ldots \times [\mathsf{GenSz}(g)]$:

1. For each party $i \in [\mathsf{GenSz}(g+1)]$ do:
   (a) Share the secret $s_{(c_0,\ldots,c_g)}$ via a $(k_{\mathsf{IdxOf}(G+1,i)} - \sum_{i=1}^{g} c_i)$-out-of-$i$ to get shares $\Pi_1, \ldots, \Pi_i$.
   (b) For each $j \in [i]$, give share Π_j to the j^{th} party in the generation.

2. For each $c_{g+1} \in [\mathsf{GenSz}(g+1)]$
   (a) Sample $r_{(c_0,\ldots,c_{g+1})} \leftarrow \{0,1\}$ uniformly at random.
   (b) Share $r_{(c_0,\ldots,c_{g+1})}$ via a c_g-ouf-of-$\mathsf{GenSz}(g+1)$ scheme among the parties of the $(g+1)^{\text{th}}$ generation.
   (c) Set $s_{(c_0,\ldots,c_{g+1})} = s_{(c_0,\ldots,c_g)} \oplus r_{(c_0,\ldots,c_{g+1})}$.

For correctness we observe that if c_i parties arrive from generation i for every $i \in [g+1]$, then by the correctness of Shamir's scheme they can recover $r_{(c_0)}$, $r_{(c_0,c_1)}$ and all the way through $r_{(c_0,\ldots,c_g)}$. Assume that the present set is qualified while the most recent party is the i-th party in generation $g+1$. Moreover, assume that from the $(g+1)^{\text{th}}$ generation there are ℓ parties present from the first i parties. Since the set is qualified, $\sum_{i=0}^{g} c_i + \ell \geq k_{\mathsf{IdxOf}(G+1,i)}$. Thus, the set of parties can further recover $s_{(c_0,\ldots,c_g)}$ (again, by the correctness of Shamir's scheme). The latter is $s_{(c_0,\ldots,c_g)} = s_{(c_0,\ldots,c_{g-1})} \oplus r_{(c_0,\ldots,c_g)}$, from which we can recover $s_{(c_0,\ldots,c_{g-1})}$ (since we know $r_{(c_0,\ldots,c_g)}$). Continuing in this manner, we can compute $s_{(c_0,\ldots,c_{g-2})}$ and then $s_{(c_0,\ldots,c_{g-3})}$ until we recover $s_{(0)}$ which is equal to the secret we shared.

For security we need to show that an unqualified set has no information regarding s, the secret that was shared. The proof is by induction on the number of generations. Assume that the scheme is secure for parties coming from g generations and we will show that it is secure for parties coming from the first $g+1$ generations. The base case follows immediately from the security of Shamir's scheme. Let the dealer share the secret among the parties in the first generation. Now, we observe that what the dealer does in the remaining sharing procedure is to share $\mathsf{GenSz}(0)$ secrets among the remaining g generations with slightly modified access structures. That is, it shares the secret $s_{(i)}$ for $i \in [\mathsf{GenSz}(0)]$ according to the sequence of dynamic thresholds $k_1 - i, k_2 - i, \ldots,$. We claim that the remaining satisfies one of two cases: (1) it is unqualified in the new access structure and therefore its shares are independent of $s_{(i)}$, or (2) it is qualified so can learn $s_{(i)}$ but in this case it won't be able to recover the masking of s (by the security Shamir's scheme). The third option where it is both qualified and can learn the masking of s cannot occur since the set is unqualified to begin with.

Now, we apply the induction hypothesis and get that the shares held by the adversary according to each of these schemes are independent of the secret. Moreover, the sharing is done independently among these access structures and therefore the combination of all of these shares is independent of the secret.

The share size. The share size of a party in generation g consists of two parts corresponding to the above two Shamir sharing procedures. The first part, stemming from Item 1 above, is of size at most

$$\prod_{j=1}^{g} \mathsf{GenSz}(j) \cdot \log(\mathsf{GenSz}(g)) = \prod_{j=1}^{g} 2^{2^j} \cdot 2^g = 2^{\sum_{j=1}^{g} 2^j} \cdot 2^g \leq 2^{2^{g+1}} \cdot 2^g.$$

The second part, stemming from Item 2 above, is (again) of size at most

$$\prod_{j=1}^{g} \mathsf{GenSz}(j) \cdot \log(\mathsf{GenSz}(g)) \leq 2^{2^{g+1}} \cdot 2^g.$$

In total, the share size is bounded by $2^{2^{g+1}} \cdot 2^g \cdot 2$. The t-th party is in generation $g = \lceil \log \log t \rceil$ which means that its share size is bounded by $4t^4 \cdot \log t$. ■

On our generation size. The choice of parameters where generation sizes grows as $\mathsf{GenSz}(g+1) = (\mathsf{GenSz}(g))^2$ were carefully chosen to obtain optimal share complexity. The "generation-like" schemes of [14] were always growing by a linear factor and such choice in our case results with an inefficient scheme in which shares are of super-polynomial size. Specifically, our goal is to minimize the value of the product:

$$\prod_{j=1}^{g} \mathsf{GenSz}(j) \cdot \log(\mathsf{GenSz}(g)).$$

Choosing generations of linearly growing size gives that $\mathsf{GenSz}(j)$ is roughly 2^j (which is indeed small for the t-th party which is in generation roughly $\log t$) but there are now logarithmically many terms in the product which results with super-polynomial share size. A further inspection gives that our choice of the constant 2 in the exponent gives the best share size.

On sharing longer secrets. The above scheme can be generalized to support sharing of longer secrets more efficiently than sharing it bit by bit. 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. More precisely, Shamir's scheme allows to share a secret of length ℓ with shares of size $\max\{\ell, \log q\}$ (where $q > n$ is a prime number as above and n is the number of parties among which we share the secret). So, even for long secrets, for large enough party index $t \in \mathbb{N}$, we will apply Shamir's scheme on a very large set such that $\max\{\ell, \log q\} = \log q$ and therefore the analysis from above will hold. For parties with low index (where $\max\{\ell, \log q\} = \ell$) we do pay a price proportional to ℓ in the share size.

3.1 A General Framework

Our scheme is a special case of the following approach that can be used for more general evolving access structures. These access structures have the property that (1) parties can be split into generations of growing size, where the size of generation g is denoted by $\mathsf{GenSz}(g)$, (2) within each generation "not too much" information has to be remembered for the future, and (3) it is possible to efficiently "combine" all this information from different generations and decide whether a set is qualified or not.

The access structure at time $t \in \mathbb{N}$, denoted by $\mathcal{A}_t$, is a function of indicator bits representing the presence of each party in the reconstruction process. Namely, we can think of the function $\mathcal{A}_t(x_1, \ldots, x_t)$ as the indicator function of the access structure (where each x_i indicates whether the i^{th} party is present). Denote by X_g the set of parties in generation g. Associate with each generation g, monotone functions $\Psi_0^g, \ldots, \Psi_{\ell_g}^g : \{0,1\}^{X_g} \to \{0,1\}$ that gets the indicator of the parties in the generation and output one bit (where ℓ_g is a parameter). Moreover, for each $(c_0, \ldots, c_{g-1}) \in \{0,1\}^{\ell_0} \times \ldots \times \{0,1\}^{\ell_{g-1}}$, associate a monotone function $\Phi_{c_0,\ldots,c_{g-1}} : \{0,1\}^{X_g} \to \{0,1\}$ such that the indicator of a set of parties $x_1, \ldots, x_t$ (where the generation of party t is g^*) is qualified in $\mathcal{A}_t$ iff

$$\begin{aligned} &\mathcal{A}_t(x_1, \ldots, x_t) = 1 \iff \\ &\exists c_0, \ldots, c_{g^*-1} \in [\ell_0] \times \ldots \times [\ell_{g^*-1}] \colon \Phi_{\Psi_{c_0}^0(X_0),\ldots,\Psi_{c_{g^*-1}}^{g^*-1}(X_{g^*-1})}(X_{g^*}). \end{aligned} \tag{3.1}$$

Such an association always exists by setting each Ψ_i' to be the identity function that outputs the ith bit (i.e., $\ell_g = \mathsf{GenSz}(g)$) and letting $\Phi_{c_0,\ldots,c_g}$ correspond to $\mathcal{A}_t$ (for the appropriate value of t) where the output of Ψ_i for each $i \in [g-1]$ is fixed and only the last generation is not. In some cases, however, there is a more efficient mapping. For example, in the dynamic threshold considered above, we set each Ψ_i to count how many parties come from that generation, namely, $\ell_i = \mathsf{GenSz}(i)$, and the monotone function $\Phi_{\ell_0,\ldots,\ell_g}$, on input $x_1, \ldots, x_{\mathsf{GenSz}(g)}$ is naturally defined to be the one that checks for each $j \in [\mathsf{GenSz}(g)]$ whether $\sum_{i=0}^{g} \ell_i + \sum_{i=1}^{j} x_i$ is at least as large as the required threshold.

The point in making the above mapping is that now the original access structure $\mathcal{A}$ can be viewed as a composition of many access structures of the form $\Psi_{c_i}^g$ and $\Phi_{c_o,\ldots,c_g}$. If we choose the generations to be large enough but keep the ℓ_i's not too large, and moreover have efficient schemes for the above structures, we can overall have an efficient scheme. We describe this general scheme next. The state of the dealer after generation g ends consists of strings s_A, where A ranges over all tuples $(c_0, \ldots, c_g)$ such that $c_i \in \{0,1\}^{\ell_i}$. Denote by $s \in \{0,1\}$ the secret to be shared and set $s_{(0)} = s$. When the $(g+1)$-th generation begins, the dealer does the following for every $(c_0, \ldots, c_g) \in [\ell_0] \times \ldots \times [\ell_g]$:

1. Share the secret $s_{(c_0,\ldots,c_g)}$ via a $\Phi_{c_0,\ldots,c_g}$ among the parties in generation $g+1$.
2. For each $c_{g+1} \in [\ell_{g+1}]$
   (a) Sample $r_{(c_0,\ldots,c_{g+1})} \leftarrow \{0,1\}$ uniformly at random.
   (b) Share $r_{(c_0,\ldots,c_{g+1})}$ via a $\Psi_{c_{g+1}}^{g+1}$ among the parties of generation $g+1$.
   (c) Set $s_{(c_0,\ldots,c_{g+1})} = s_{(c_0,\ldots,c_g)} \oplus r_{(c_0,\ldots,c_{g+1})}$.

The correctness and security of the scheme follows by identity 3.1, similarly to how we proved correctness and security for the dynamic threshold scheme. We omit further details here.

The share size of a party in generation $g+1$ consists of two parts corresponding to the above two Φ and Ψ sharing procedures. We assume that the share size of each $\Phi_{c_0,\ldots,c_g}$ upper bounded by $\phi_{c_0,\ldots,c_g}$ and that the share size of each $\Psi_{c_g}^g$

is upper bounded by $\psi^g_{c_g}$. The first part, stemming from Item 1 above, is of size at most

$$\prod_{c_0 \in [\ell_0]} \cdots \prod_{c_g \in [\ell_g]} \phi_{c_0,\ldots,c_g}.$$

The second part, stemming from Item 2 above, is of size at most

$$\prod_{c_0[\ell_0]} \cdots \prod_{c_g \in [\ell_g]} \prod_{c_{g+1} \in [\ell_{g+1}]} \psi^g_{c_g}.$$

In total, the share size of party t that resides in generation g is bounded by the sum of the two terms above.

Instantiations. The above general blueprint captures not only the dynamic threshold scheme we presented above, but also can be used to capture the scheme for general access structures and the scheme for k-threshold for constant values of k of [14]. However, the choice of the generation size is different in each case. In the general case, the generations are of size 1 (as we cannot gain anything from squashing since the structure is completely arbitrary), and in the k-threshold case, the generations are growing in *linear* rate (linear in k) rather than polynomial in t as we have in the dynamic threshold case.

4 Robust Evolving Secret Sharing

In this section we show how to generically make any k-threshold scheme robust in the sense that even if some parties hand-in incorrect shares, the correct secret can be recovered. The formalization of this notion is done by augmenting a standard secret sharing for evolving access structures with an additional procedure called $\mathsf{R\text{-}RECON}$ which gets as input the shares of a set of parties A from which it can recover the secret. The adversary is allowed to corrupt any set $B \subseteq A$ such that $A \backslash B$ is still qualified. The aforementioned reconstruction procedure succeeds with all but $2^{-\lambda}$ probability, where λ is a parameter that is fixed during the sharing procedure.

Definition 4.1 (Robust evolving secret sharing). *A robust secret sharing scheme $\mathcal{R}$ is described by three procedures* $(\mathsf{SHARE}, \mathsf{RECON}, \mathsf{R\text{-}RECON})$. *The procedures* $(\mathsf{SHARE}, \mathsf{RECON})$ *form an evolving secret sharing scheme (as in Definition 2.4) in which the procedure* SHARE *is augmented with an additional input* 1^λ *for a security parameter* λ. *The additional procedure* $\mathsf{R\text{-}RECON}$ *satisfies the following requirement:*

3 ***Robust reconstruction:*** *The secret s is shared using* $\mathsf{SHARE}(1^\lambda, s)$. *An adversary $\mathcal{A}$ chooses a time t and two subsets of parties $A, B \subseteq [t]$ such that (1) $B \subseteq A$, (2) B is unqualified, and (3) $A \backslash B$ is qualified. The adversary $\mathcal{A}$ is then given the shares of the parties in B, denoted by Π^s_B, and it changes it arbitrarily to get $\Pi^s_B{}'$. Finally, the value of $s' = \mathsf{R\text{-}RECON}(1^\lambda, \Pi^s_A \cup \Pi^s_B{}')$ is output.*

We say that the scheme is λ-robust if for any such adversary $\mathcal{A}$ if it holds that

$$\Pr[s' \neq s] \leq 2^{-\lambda}.$$

The next theorem shows how to obtain a robust secret sharing scheme for the evolving k-threshold access structure in which qualified sets are those of size at least k.

Theorem 4.2. *Let $k \in \mathbb{N}^+$ and $\lambda > 0$. Assume there exists a linear evolving (family of) schemes for k-threshold such that for the domain of secrets $\mathcal{S}$, it is linear over the field $\mathbb{F} = \mathbb{F}_2^t$ ($t \geq \log |\mathcal{S}|$).*

Then, there exists an evolving λ-robust secret sharing scheme for the evolving k-threshold access structure. The overhead in the share for party t is an additive factor of $O(\lambda + k \cdot \log k)$ bits relatively to the share size of the original scheme (for a sufficiently large domain $\mathcal{S}$, otherwise the overhead is multiplicative).

We prove the theorem, by adapting the robust (standard) secret sharing scheme of [10] to the evolving setting. Then, we use the linear scheme of [14] for the evolving k-threshold access structure and transform it into a robust one.[3] The high-level idea of the construction is, instead of sharing the secret itself, to share an AMD encoding of the secret (see Definition 2.6). Roughly speaking, the resulting scheme is robust since AMD codes protect information against additive attacks and our secret sharing scheme is linear.

Proof of Theorem 4.2. Our construction assumes a linear evolving scheme $\mathcal{E} = (\mathsf{SHARE}, \mathsf{RECON})$ for a k-threshold access structure and turns it into a robust evolving scheme for the same structure. We share secrets from domain $\mathcal{S}$. As an instantiation of the base scheme, we use the construction from [14] for the evolving k-threshold access structure over a sufficiently large secret space. The share size for the t-th party in their scheme is roughly $\sigma(t) = k \log t$ bits for large enough t. Fix a $\gamma' = (\lambda + k \log k)$-AMD code (E, D) for secret domain $|\mathcal{S}|$. Concretely, we use $\mathsf{AMD}_{\sigma,\gamma'}$.

Our new robust secret sharing scheme is described next:

1. The new sharing procedure $\mathsf{SHARE}'(1^\lambda, \Pi_1^s, \ldots, \Pi_{t-1}^s, s)$ gets as input a robustness parameter 1^λ, the shares of parties $1, \ldots, t-1$ and the original secret s and generates the share for the t-th party as follows. At the beginning of time (before the first party arrives), it computes an AMD encoding of s, denoted $\hat{s} = E(s)$, and shares this value using the underlying scheme by running (in the t-th time step) the procedure $\mathsf{SHARE}(\Pi_1^s, \ldots, \Pi_{t-1}^s, \hat{s})$ and giving the t-th party this value.

[3] Observe that the construction from [14] for the evolving k-threshold access structure are "almost" of the right form. One minor issue is that the field over which the various instances of Shamir operate grow as more parties arrive. Using extension fields, the shares can be viewed as a vector of linear combinations over a single field $\mathbb{F}_2^t$ of a suitable size, and the proof applies in a similar way. Our scheme from Theorem 3.3 has the same property.

2. The reconstruction procedure $\mathsf{RECON}'(\Pi^s_B, B)$ on input the shares of a subset of parties B applies the original reconstruction procedure of the underlying scheme $\mathsf{RECON}(\Pi^s_B, B)$ to obtain an AMD encoding $\hat{s}$. Then, it outputs the AMD decoding of this value $s = D(\hat{s})$.
3. The robust reconstruction procedure $\mathsf{R\text{-}RECON}(1^\lambda, \Pi^s_B, B)$ on input the robustness parameter 1^λ and the shares of a set of parties B works as follows. Let B' denote the set of the first $\min\{2k-1, |B|\}$ parties in B. Go over all minterms $T \subseteq B'$ (sets of size exactly k), and apply the reconstruction procedure on each of them: $\hat{s}_T = \mathsf{RECON}'(1^\lambda, \Pi^s_T, T)$. If all $\hat{s}_T$ are $\perp$, output $\perp$. Otherwise, output the first value which is not $\perp$.

   Notice that since k is constant, the running time of this procedure is polynomial in its input size.

We proceed with the correctness, security and robustness of the above construction. As the original scheme is an evolving k-threshold scheme, and as the AMD scheme is perfectly correct the resulting scheme satisfies perfect correctness and privacy. As to robustness, first observe that $|B'| \leq 2k-1$, and it must contain a qualified subset T' in which no party is malicious. Indeed, if $B' = B$, this follows by our guarantee on the choice of the malicious parties the adversary is allowed to make (otherwise, the adversary chose a qualified set which is illegal). If $|B'| = 2k-1$, then the set of honest parties in this subset is of size k, and is therefore qualified.

Next, we prove that with probability at least $1 - 2^{-\lambda}$, the robust reconstruction procedure $\mathsf{R\text{-}RECON}$ outputs the shared secret s. By perfect correctness of the AMD scheme, $\hat{s}'_{T'} = s$. It remains to show that for all other minterms T, it holds that $\hat{s}_T \in \{s, \perp\}$ with probability $1 - 2^{-\lambda}$ (the proof is similar to the one in [10], and is included here for completeness). For each T, consider any possible shift Δ_T in the shares chosen by the adversary. This shift naturally corresponds to an additive shift on the total set of shares used for reconstruction, as thus on the shared value (since the scheme basic evolving k-threshold scheme is linear).

By the security of the secret sharing scheme, the adversary's view (i.e. the shares of the parties he controls) does not depend on $\hat{s}$. Thus the distribution of shifted shares is also independent of the secret $\hat{s}$. Now, by the security of the AMD code, $\hat{s}_T \notin \{s, \perp\}$ with probability at most $2^{-\lambda + k \cdot (\log k + 1)}$. As there are at most $\binom{|B'|}{k} \leq \binom{2k-1}{k}$ possible different sets T (minterms), we can apply a union bound and get that the probability that this happens for *some* $\hat{s}_T$ is at most

$$(2k)^k \cdot 2^{-\lambda + k \cdot (\log k + 1)} = 2^{k \cdot (\log k + 1)} \cdot 2^{-\lambda + k \cdot (\log k + 1)} \leq 2^{-\lambda},$$

as required. ■

Acknowledgments. We thank Amos Beimel and the anonymous reviewers for their comments and suggestions.

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: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 11–46. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20901-7_2
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC, pp. 1–10 (1988)
4. Bishop, A., Pastro, V., Rajaraman, R., Wichs, D.: Essentially optimal robust secret sharing with maximal corruptions. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 58–86. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_3
5. Blakley, G.R.: Safeguarding cryptographic keys. In: Proceedings of the AFIPS National Computer Conference, vol. 22, pp. 313–317 (1979)
6. Cevallos, A., Fehr, S., Ostrovsky, R., Rabani, Y.: Unconditionally-secure robust secret sharing with compact shares. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 195–208. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_13
7. Cramer, R., Damgård, I., Fehr, S.: On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 503–523. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_30
8. Cramer, R., Damgård, I., Maurer, U.: General secure multi-party computation from any linear secret-sharing scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 316–334. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_22
9. Cramer, R., Damgård, I., Nielsen, J.B.: Secure Multiparty Computation and Secret Sharing. Cambridge University Press, Cambridge (2015)
10. Cramer, R., Dodis, Y., Fehr, S., Padró, C., Wichs, D.: Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 471–488. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_27
11. Csirmaz, L.: The size of a share must be large. J. Cryptol. **10**(4), 223–231 (1997)
12. Ito, M., Saito, A., Nishizeki, T.: Multiple assignment scheme for sharing secret. J. Cryptol. **6**(1), 15–20 (1993)
13. Kol, G., Naor, M.: Games for exchanging information. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC, pp. 423–432 (2008)
14. Komargodski, I., Naor, M., Yogev, E.: How to share a secret, infinitely. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 485–514. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53644-5_19
15. Paskin-Cherniavsky, A.: How to infinitely share a secret more efficiently. IACR Cryptol. ePrint Arch. **2016**, 1088 (2016)
16. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 73–85 (1989)
17. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)