Tight Lower Bounds and New Upper Bounds For Evolving CDS

Tamar Ben David* tamarbd@ariel.ac.il

Anat Paskin-Cherniavsky* anatpc@ariel.ac.il

February 19 2025

Abstract

Komargodski et. al. defined *evolving secret-sharing schemes* with an unbounded number of parties. In this model, parties arrive one after the other and the number of parties that will arrive is not known. Another cryptographic primitive related to secret-sharing is *conditional disclosure of secrets protocols* (CDS) that defined in [21]. A CDS protocol for a Boolean function f involves k servers and a referee. Each server holds a common secret s, a common random string r, and a private input x_i; using these r, each server locally computes one message and sends it to the referee. The referee, knowing the inputs $x_1, \cdots, x_k$ and the messages, should be able to compute s if $f(x_1, \cdots, x_k) = 1$. Otherwise, the referee should not learn information about the secret. In a sense, this is a non-monotone version of secret sharing schemes.

Peter (ISC 23'), defines evolving CDS, implementing in particular evolving predicate $f : \{0,1\}^* \to \{0,1\}$ (he handles somewhat more general predicates for larger input domains, but generalizing to other input domains is not hard, and we focus on boolean predicates). In this setting, the number of parties is unbounded, where the parties arrive in sequential order. Each party, when arrives, sends one random message to a referee, that depending on its input, the secret, and a common randomness. He also devise evolving CDS protocols for a general evolving predicate via a black-box reduction to evolving secret-sharing scheme for a related access structure. He analyzes this construction for general access structures, as well as other classes of functions, which yields message complexity $O(2^t)$ for the worst predicates. In this work we provide new upper and lower bounds for evolving CDS.

- Observing that CDS has the potential for improved efficiency, as it is not restricted to monotone operations, we devise a new evolving general CDS construction. In particular, our construction relies on representing the evolving predicate via an infinite branching program - LINBP, generalizing the monotone infinite branching program based construction of Alon et. al of evolving secret sharing schemes. We obtain nontrivial ($2^{ct-o(t)}$ for $c < 1$) message complexity for branching programs of larger width than Alon et al's construction (even when restricting attention to monotone predicates), as well as Peter's construction for certain (but not all) f's.
- Indeed, we prove that our construction, as well as [30] is tight for a certain evolving predicate – as for evolving secret-sharing, (so called *strong*) evolving CDS also requires share complexity of $2^{t-o(t)}$. This is unlike the state of affairs for the finite setting, where the best known CDS constructions are much more efficient than the best known secret sharing schemes (for the hardest monotone functions). The latter bound is proved based on an adaptation of Mazor's lower bound (in turns based on Csirmaz's lower bounding

*Department of Computer Science, Ariel University, Ariel Cyber Innovation Center (ACIC).

technique) to the CDS setting. It relies on a reduction from secret sharing for a certain class of infinite access structures – the so called *partite* access structures to evolving CDS for a related (not necessarily monotone) function. Then, a partite evolving access structure is crafted using the Csirmaz-type argument.

1 Introduction

In secret-sharing schemes [11, 32, 22], there are n parties and a dealer, which holds a secret. The dealer applies a randomized algorithm to the secret, resulting in n strings called shares, and gives the i^{th} share to the i^{th} party. Only certain predefined qualified sets of parties can reconstruct the secret, while any other set gains absolutely no information about it from their shares. The collections of predefined qualified sets of parties are called access structures. Secret-sharing schemes are well-studied and have many cryptographic applications. This model assumes that the number of parties is known in advance and the share size, are determined by the number of parties.

This assumption poses problems in many scenarios. Of course, one can take some upper bound on the number of parties. However, if this bound is big, the share size will be unnecessarily large, even if only a few parties actually participate in the scheme. Conversely, if the bound is too small, there is a risk that too many parties may arrive, and no further shares can be produced; this will require an expensive re-sharing of the secret and updating all shares (which can be impossible if some parties are temporally off-line). Thus, we need to consider models with an unbounded number of parties.

Komargodski, Naor, and Yogev [24] defined *evolving secret-sharing schemes* with an unbounded number of parties. In this model, parties arrive one after the other and the number of parties that will arrive is not known. At the beginning of the execution, the dealer holds a secret (as in the common model). When a party arrives, the dealer computes a share and gives it to the party; this share cannot be updated in the future. Thus, when preparing the t^{th} share, the dealer cannot assume any bound on the number of parties that will eventually arrive; the size of the t^{th} share should be measured as a function of t. Correctness and privacy are required with respect to an *evolving access structure*, where the parties are $\{p_i\}_{i\in\mathbb{N}}$ and the evolving access structure is a collection of finite subsets of the parties that are authorized to reconstruct the secret.It is assumed that the order that the parties that arrive is known in advance (in the literature it is the natural order).

Another cryptographic primitive related to secret-sharing is *conditional disclosure of secrets protocols* (CDS) that defined in [21]. CDS can be roughly viewed as a non-monotone version of secret sharing, and has also had broad applicability in the literature. Notably, in the recent seminal breakthrough upper bounds for secret sharing with share complexity 2^{ct} for $c < 1$ [27], as well as in the currently best known secret sharing with share complexity 1.5^n, based on the so called robust CDS [3]. A CDS protocol for a Boolean function f involves k servers and a referee. Each server holds a common secret s, a common random string r, and a private input x_i; using these r, s, and x_i the i-th server computes one message (without seeing any other input or message) and sends it to the referee. The referee, knowing the inputs $x_1, \cdots, x_k$ and the messages, should be able to compute s if and only if $f(x_1, \cdots, x_k) = 1$. Otherwise, the referee should not learn information about the secret. CDS protocols have many cryptographic applications and one of them is to reduce the share size of secret-sharing schemes for general access structures[27, 8, 7].

Motivated by similar goals to those of studying evolving secret sharing [30] defines evolving CDS.

In this setting, the number of parties is unbounded, and the parties arrive one by one in sequential order. Each party, when arrives, sends one message to a referee, which is never changed in the sequel. Evolving CDS protocols implement so called evolving predicates $f : \bigcup_{i=1}^{\infty} X_1 \times \ldots \times X_i$. This sequence of predicates is monotone, in the sense that if a predicate f_t holds on some input $\mathbf{x} = (x_1, \ldots, x_t)$ f_t, then f_{t+1} holds on any input of which $\mathbf{x}$ is a prefix. Indeed, such monotonicity is necessary, as once $f_t(x_1, \ldots, x_t) = 1$, and the secret is revealed by messages $m_{1,x_1}, \ldots, m_{1,x_t}$, in must be valid to reveal it based of any input $\mathbf{x}'$ of which $(x_1, \ldots, x_t)$ is a suffix (as messages are never discarded by the referee).

1.1 Prior Work

1.1.1 Evolving Secret-Sharing Schemes.

We next briefly discuss the known results on evolving secret-sharing schemes. Komargodski et al. [24] showed that every monotone evolving access structure can be realized by an evolving secret-sharing scheme; in this scheme the size of the t^{th} share is 2^{t-1}. Mazor [28], quite surprisingly, proved that this is almost tight. In particular, for every $g(t) \in \omega(t)$, there exists an evolving access structure for which any evolving secret sharing scheme requires $\Omega(2^{t-g(t)})$ (total) share complexity for all t. This is to contrast with the best lower bound for finite secret sharing schemes of $O(n/log(n))$ [15], which leaves an exponential gap between best known lower and upper bounds for the 'hardest' evolving access structures. On the positive side, Komargodski et al. and follow-up works [25, 17, 5, 18, 6, 13, 16, 20, 29, 31, 33, 34, 30, 1, 19, 14] constructed efficient evolving secret-sharing schemes for natural classes access structures.

In light of Mazor [28], [1] have searched for as rich as possible a class of evolving access structures that can be realized with an evolving secret-sharing scheme that has non-trivial share size; (e.g, schemes with share size 2^{ct}). As many secret-sharing schemes for *finite* access structures use a representation of the access structure by some computational model to construct a secret-sharing scheme realizing the access structure, e.g, CNF and DNF formulas are used in [22], monotone formulas are used in [10], and monotone span programs are used in [23], they define a certain computational model, for which they construct suitable evolving secret-sharing schemes. In some more detail, their construction abstracts and generalizes the constructions of [24, 25], defining layered monotone infinite branching programs (LIBP, which are ordered and read-once), which represent evolving access structures. They transform an LIBP into an equivalent simpler computation model they call generalized infinite decision trees (abbreviated GIDT), and show how to construct evolving secret-sharing schemes for generalized infinite decision trees. In particular, the resulting evolving secret-sharing schemes has non-trivial share complexity for width $w(t) = 2^{0.15t}$ (this is close to, but not completely tight, resulting in share complexity of $\leq 2^{0.97t}$).

1.1.2 Evolving Conditional Disclosure of Secrets

As mentioned above, recently Peter [30] formalized evolving conditional disclosure of secrets (CDS) protocols. The paper also includes a proof that all evolving predicates have evolving CDS protocols, with complexity $O(2^t)$, similarly to the setting of evolving secret sharing, as well as improved complexity for certain classes of evolving predicates, such as evolving minsum predicates. His construction for f is based on a black box reduction to the best known evolving secret sharing scheme for an access structure $\mathcal{A}_f$ related to f. In particular, it implicitly implies the existence of an evolving CDS for f for which $\mathcal{A}_f$ has an LIBP with relatively small width (say $O(2^{0.15t})$.

1.2 Our Results

We revisit the question of the message complexity of evolving CDS, making progress on two open questions, as we see them, posed by the work [30].

Question 1. Are there more efficient constructions for broad classes of functions¿ In particular those with LIBP's with small width (that need not be non-monotone, in fact)?

This question makes sense, as [30]'s construction reduces evolving CDS to evolving secret sharing, while in the finite setting reductions usually go in the other direction, and often the best known constructions for CDS are better than the best known secret sharing constructions (see more below)

Question 2. Is [30]'s construction tight in the worst case?

Indeed, we answer the first question in the affirmative (in terms of the best known constructions). Informally, our main construction yields the following upper bound.

Theorem 1.1 (Informal)**.** *Let $f : \{0,1\}^* \to \{0,1\}$ denote an evolving predicate, that has a (non-monotone) infinite layered branching program of width upper bounded by $w(t) = 2^{ct}$, where $0 < c < 0.25$ is a constant. Then there exists an evolving CDS protocol for f with message complexity $2^{dt+o(t)}$ where $0 < d < 1$ is a constant (that depends on c).*

This is a strict improvement over a similar evolving secret sharing construction which is based on a representation of the AS via (infinite) monotone branching programs, where share complexity is non-trivial only for BP width of (slightly rounding down the constant) $2^{0.15t+o(t)}$.

Roughly speaking, an improvement was to be expected, as unlike secret sharing, CDS is not restricted to monotone operations. Indeed, in the finite setting the best known general CDS protocols for the hardest functions are strictly more efficient than their secret sharing counterparts, for functions of the same domain $f : \{0,1\}^n \to \{0,1\}$ ($2^{\tilde{O}(n^{0.5}}$ vs $2^{O(n)}$) share complexity. This holds even for monotone f's, for which both models are applicable.

As a corollary, we improve over [1] for a certain class of evolving secret sharing schemes – the so called *partite* access structures (see Section 5 for further discussion). In a nutshell, a partite access structure $\mathcal{A}_f$ corresponding to an evolving predicate $f : [N] \to \{0,1\}$, say, is defined over the set of parties $\{p_(i,v)\}_{i\in\mathbb{N},v\in[N]}$, and the sets containing some $\{p_{1,v_1},\ldots,p_{t,v_t}\}$ where $f(v_1,\ldots,v_t) = 1$ or a pair $\{p_{i,v}, p_{i,w} | v \neq w\}$. The improvement follows by observing, that similarly to the finite setting, one can easily demonstrate potentially improved constructions for partite access structures $\mathcal{A}_f$ by a reduction to evolving CDS for f.

Theorem 1.2 (Informal)**.** *Let $f : \{0,1\}^* \to \{0,1\}$ denote an evolving predicate for secret domain S, which has a strong CDS with message complexity $emc(t)$. Then $\mathcal{A}_f$ has an evolving secret-sharing scheme with share complexity $emc(t/2) + max(\log(|S|), \log n + 1)$.*

More precisely, unlike in the finite setting, we need the CDS to be strong. That is, if no contiguous set of servers with a joint input $\mathbf{v} = (v_1,\ldots,v_t)$ where $f(\mathbf{v}) = 1$ shows up, then nothing is learned about the secret. Unlike the finite setting, where strong CDS can be reduced to CDS with a very small message complexity overhead, we do not generally know how to make evolving CDS strong with a small overhead (we only know how to directly construct such CDS). See Section 4 for more discussion.

On the negative side, we answer Question 2 in the positive, falsifying the hope of obtaining strong CDS protocols with non-trivial efficiency.

Theorem 1.3 (Informal)**.** *There exists an evolving predicate $f : \{0,1\}^* \to \{0,1\}$, such that any strong evolving CDS for it, and secret domain $S = \{0,1\}$ does not have a message complexity bound of the form $st(t) = O(2^{ct})$ for a constant $c < 1$.*

Removing the caveat that the lower bound is only for strong CDS is an interesting open problem. Finally, we observe that our results focus on the boolean input domains merely for convenience, and our techniques readily generalize to arbitrary input domains.

1.3 Our Techniques

1.3.1 Upper Bounds

1.3.1.1 An IBP-based CDS construction, and comparison to IBP-based evolving secret sharing schemes.

The construction is quite similar in spirit to the evolving secret sharing construction of [1]. We assume familiarity with the construction of [1] for the sake of the exposition here (the technical sections of the paper are self contained). In a nutshell, similarly to their construction, our construction for an evolving predicate $f : \{0,1\}^* \to \{0,1\}$ relies on representing f via infinite branching programs (LINBPs) and (generalized) infinite decision trees (GINDTs). In some more detail, an LINBP is a natural infinite branching program computation model for evaluating evolving predicates. It generalizes the LIBP introduced in [1], by allowing nonmonotone edge labeling predicates. A GINDT is a natural infinite tree computation model for evaluating evolving predicates. A (layered) graph of a LINBP B is specified by an infinite rooted DAG $G(V, E)$, partitioned into (finite) layers, $L_0, L_1, \ldots$, where L_0 consists of the root. Every edge $e \in L_i \times L_{i+1}$ is labeled by a predicate μ_e in the variable x_{i+1}. Naturally, a vector $\sigma^t =\in \{0,1\}^t$ is accepted by B if it satisfied all predicates along a path from the root to a leaf. As for evolving secret sharing, we do not directly know how to use an LINBP to build an evolving CDS protocol, as in [1], thus we transform it into an equivalent generalized infinite decision tree GINDT instance. We partition the variables in the LINBP into consecutive sets called generations. In The generations are defined by some function $h : \mathbb{N} \to \mathbb{N}$; the i^{th} generation contains the variables $x_{h(i-1)+1}, \ldots, x_{h(i)}$. An infinite non-monotone decision tree $T(G, h, v_0, \mu)$ generalizes LINBP in that the predicates labeling labeling an edge e going from layer T_i to layer T_{i+1} of the tree depend on several variables - those in generation $i+1$. The generations are determined by the function $h : \mathbb{N} \to \mathbb{N}$; the i^{th} generation contains the variables $x_{h(i-1)+1}, \ldots, x_{h(i)}$. It is also restricts LINBP in the sense that G has a tree structure. Similarly to the IBP to GIDT tranasformation in [1], the GINDT model is inspired by an optimized transformation of LINBP into an equivalent GINDT instance. Next, we describe the transformation, while emphasizing the differences between it and the transformation from LIBP into GINDT in [1]. In T, there is a vertex $u_{j_1,\ldots,j_i}$ for any sequence of vertices $u_{j_1}, \ldots, u_{j_i}$ in layers $h(1), \ldots, h(i)$ respectively, which contain no leafs. We connect a vertex representing a path p with a vertex of the form (p, v) for some $v \in B$. Every such $e = (u_{p,v'}, u_{p,v',v})$ for v' in layer $h(i-1)$ in B is labeled by a predicate μ_e in the i'th generation variables, that states that some path from v' to v in B is satisfied by this set of variables. Also, we add a leaf $u_{j_1,\ldots,j_{i-1},j}$ for every layer $h(i-1) < j \leq h(i)$ in B. The leaf $u_{j_1,\ldots,j_{i-1},j}$ is the child of $u_{j_1,\ldots,j_{i-1}}$. The predicate μ_e of $e = (u_{j_1,\ldots,j_{i-1}}, u_{j_1,\ldots,j_{i-1},j})$ accounts for all paths from $u_{j_{i-1}}$ to some leaf in layer j in B. This part of the tree differs from the [1] conversion, which only had a single leaf $v_{j_1,\ldots,j_{i-1}}$ descending from every $u_{j_1,\ldots,j_{i-1}}$, which represented all paths from $u_{j_1,\ldots,j_{i-1}}$ to some leaf in layer $h(i-1) < j \leq h(i)$. This change is required due to

the technical restriction that (standard) CDS needs all servers in its predefined server set to send a message corresponding to a certain input for the correctness guarantee to hold. In particular, a predicate labeling $e = (u_p, u_{p,j})$ depends only on variables $x_{h(i-1)}, \ldots, x_j$, instead of all the variables in generation i. Secret-sharing for parties $\{p_1, \ldots, p_n\}$, on the other hand, can seamlessly handle a qualified subset's $S \subseteq \{p_1, \ldots, p_{n/2}\}$ shares, say, as shares of a subset of $\{p_1, \ldots, p_{n/2}\}$ [1] [2]. See Section 4 for a formal definition of these computational models. To construct a CDS protocol from $T(G = (V, E), u_0, \mu)$ for $f : \{0,1\}^* \to \{0,1\}$, we proceed in two steps. First, we turn T into an IDT T' (which is a special case of IDT), by modifying its predicates and set of variables. Briefly, every edge e in an IDT is assigned a predicate y_e - a unique variable. The outer scheme is an evolving *secret sharing* scheme for the IDT (where qualified sets are those satisfying all predicates on a path from the root to a leaf). It can be implemented by an ideal evolving secret sharing [24, 1]. Then, for each $e = (u_{p,v'}, u_{p,v',v}) \in E$ where v', we apply a (finite) CDS for predicate $\mu(e)$ for y_e's share sh_e as the secret among the set of parties on which $\mu(e)$ depends (either an entire generation or a prefix thereof). We then append the corresponding messages $m_{i,0}$ ($m_{i,1}$) to q_i's 0 (1) corresponding CDS messages. Overall, the message complexity of the protocol is $O(h(\ell_t)\Pi_{i=1}^{\ell_t} w(h(i))$ executions of CDS protocols, each among (up to) $h(\ell_t)$ parties for a predicate of and an edge leaving layer $i-1$ in T.

We stress that although we build an evolving CDS, the outer scheme is still a secret sharing scheme for a monotone access structure, implemented among a set of virtual parties, corresponding to edges of an IDT. The main source of improved efficiency here is the fact we can use CDS protocols instead of secret sharing protocols for the edge predicates. This is an improvement of $2^{\tilde{O}(n^{0.5})}$ vs $2^{O(n)}$ per edge, as discussed above. In particular, unlike the [1] construction, for branching programs of exponential width, the cost of individual CDS's is absorbed in the $o(t)$ part in the exponent of Theorem 1.1.

Another potential source of efficiency improvement, could be the fact that we use a non-monotone branching program - an LINBP rather than an LIBP, which must have monotone predicates. However, an LINBP for f, has essentially the same structure as an LIBP for $\mathcal{A}_f$, which is the corresponding partite evolving access structure, eliminating this factor as potential improvement. In turn, evolving CDS for f can be reduced to evolving secret sharing for $\mathcal{A}_f$ in a black box manner, and almost no overhead (as done in [30]).

Finally, to make the CDS strong, it turns out that using a strong CDS variant for the edges results in a strong CDS construction.[3]

Most examples of monotone functions where there is a gap between the best known the best known evolving secret sharing and CDS schemes is substantial have exponential BP width, as in those the saving of CDS vs secret sharing for edge predicates is most substantial. Indeed, this is the case resulting from analyzing our main construction specified in Theorem 1.1. However, our approach could provide improved evolving CDS (over evolving secret sharing) complexity for

[1] We could have also handled this issue at the level of evolving CDS construction for the GINDT, but it would slightly complicate the implementation of the scheme, which we preferred to avoid.

[2] Note that monotonicity of f in the sense of Item 1 of Section 2.2.1 is implied by the definition of the LINBP and GINDT models, as acceptance is defined by satisfying a path from root to leaf (which accounts for a finite prefix of the variables). So, this restriction is required to allow the function to be computed by natural computational models, not just for the evolving CDS setting to be applicable.

[3] Note the subtlety that even 1-predicates receive outer-scheme shares, to make sure that all servers in that generation show up with some value. Otherwise, the overall protocol does not necessarily end up strong in the sense we require.

simpler functions as well (for which $\{f_t\}_{t\in\mathbb{N}}$ is in $NC1$).

Example 1.4. A finite weighted threshold (WT) function $f : \{0,1\}^n \to \{0,1\}$ is specified by vector $\mathbf{w} \in \mathbb{N}^n$ and number $\theta \in \mathbb{N}$, with qualified sets specified by vectors $< w, \mathbf{x} > \geq \theta$. It is known that for any WT function has a best known secret-sharing scheme with complexity $n^{O(\log n)}$ [9], while every such function has a (non monotone) formula of polynomial size (and thus a polynomial CDS), for a certain worst case setting of $\mathbf{w}, \theta$. Now, consider a function $f : \{0,1\}^n \to \{0,1\}$ of the form $\bigvee_{j=1}^{\infty} f_j(x_{I_j})$, where f_j is a 'worst case' WT access structure on the set of variables I_j. The consecutive intervals I_j are a partition of $\mathbb{N}$, where $|I_j| = 2^j$. To cast in our framework, this access structure may be represented directly via a GIDT $T(G(V,E), v_{0,1}, \mu)$ with 2 vertices in all layers $i > 0$, $L_i = \{v_{i,1}, v_{i,2}\}$. $v_{i,2}$ is a leaf. For all $i \geq 0$, there is an edge $e_i = (v_{i,1}, v_{i+1,2})$ labeled by $\mu_{e_i} = f_i$, and $e'_i = (v_{i,1}, v_{i+1,1})$, $\mu_{e'_i} = 1$. The message complexity of the evolving CDS resulting from the GIDNT and the evolving secret-sharing scheme resulting from the GIDT are $t^{O(1)}$ and $t^{O(\log t)}$ respectively, similarly to the finite setting.

1.3.1.2 Comparison with [30]'s work.

The construction of [30] is a black box reduction of evolving CDS for an evolving predicate $f : \{0,1\}^* \to \{0,1\}$ to its partite access structure counterpart $\mathcal{A}_f$. As mentioned above, an instantiation of $\mathcal{A}_f$ with the [1] construction, can use an LIBP that essentially mirrors the best LINBP for f. Thus, the only reason our is more efficient than [30] for some f's, is because the (monotone) edge predicates in [1] are implemented via secret-sharing, rather than via CDS, which can use non-monotone operations, and thus do it more efficiently. Also, observe that the resulting CDS is not strong, as partite functions reveal the secret whenever $p_{i,0}, p_{i,1}$ are both present. In fact, Theorem 1.2 is based on a simple reduction in the other direction - from evolving secret sharing for evolving partite access structures $\mathcal{A}_f$ to strong CDS for f. Plugging in our LINBP-based construction for stong CDS, may improve over [1]'s construction based on that LIBP because of potentially improved edge predicates in the edge CDS protocols (note that the CDS protocols can never be worse than secret sharing for $\mathcal{A}_f$, as CDS for f can always be impelemnted based on secret-sharing for $\mathcal{A}_f$ in the finite setting as well).

1.3.2 Lower bounds

In this section, we demonstrate that similarly to evolving secret-sharing schemes, (strong) evolving CDS protocols can not be achieved with non-trivial message complexity of $2^{ct+o(t)}$ for constant $c < 1$. We prove the result in two steps. First, we craft a partite evolving secret-sharing scheme $\mathcal{A}_f$ for which we can prove a tight lower bound on share complexity. The structure of $\mathcal{A}_f$ is inspired by Mazor's evolving secret-sharing scheme [28] - see Remark 1.7 for more discussion. Then, we use Theorem 5.1 to derive a lower bound for strong evolving CDS.

Theorem 1.5 (Informal)**.** *There exists an evolving predicate $f : \{0,1\}^* \to \{0,1\}$, such that any strong evolving CDS for it, and secret domain $S = \{0,1\}$ does not have a message complexity bound of the form $st(t) = O(2^{ct})$, where $c < 1$ is a constant.*

As in most secret-sharing lower bounds for general schemes, we rely on an adaptation for evolving secret-sharing schemes of the seminal technique by Csirmaz [15, 12] for finite access structures. This technique has led to the best known lower bounds for worst case access structures in the

finite setting [15], and best known bounds for concrete access structures such as st-conn [1] and k-hypergraphs [4], to name a few recent results. It also led to the best known (tight) lower bounds for the worst evolving secret-sharing schemes [28], as well as evolving variant of st-conn and evolving k-hypergraphs [1]. At the center of the adapted technique is a notion of so called *Independent Sequences*.

Definition 1.1 (Independent Sequences [12, 1])**.** Let $n, \ell \in \mathbb{N}$ be integers, let $B = \{p_1, \ldots, p_\ell\}$, $A \subseteq \{p_{\ell+1}, \ldots, p_n\}$ be a set of parties, and let $\mathcal{A}$ be an access structure whose set of parties is $\{p_1, \ldots, p_n\}$. An *independent sequence of length ℓ with respect to A* is a sequence $A_1, \ldots, A_\ell \subseteq A$ of subsets of A such that the following holds.

1. $\{p_1, \ldots, p_i\} \cup A_i \in \mathcal{A}$ for all $i \in \{1, \ldots, \ell\}$.
2. $\{p_1, \ldots, p_{i-1}\} \cup A_i \notin \mathcal{A}$ for all $i \in \{1, \ldots, \ell\}$.

Theorem 1.6 ([12, 15])**.** *Let Γ be an access structure whose set of parties is $P = \{p_1, \ldots, p_n\}$. Assume there exists $\ell \in [n]$ and $A \subseteq \{p_{\ell+1}, \ldots, p_n\}$, for which there exists an independent sequence of length ℓ with respect to A. Then for every secret-sharing scheme that realizing Γ, the total share size of the parties in A is at least $\ell - 1$.*

In some more detail, [28] builds an AS $\mathcal{A}$ in a way that Theorem 1.6 can be applied to an infinite sequence of finite (A^i, B^i) subsets of parties, so that the required lower bound holds. Here the B^i's are disjoint sets, which partition a 'sparsely located' infinite subset $B \subseteq \mathbb{N}$. A^i consists of the first i elements of $\mathbb{N} \setminus B$. By establishing an independent sequence of 2^i minterms (a convenint way to satisfy items 1,2 in Definition 1.1) of $A^i \cup B^i$ with respect to A^i, he concludes that, in any evolving scheme, the shares that the parties in A^i receive are of total size at least $2^i - 1$. The disjointness of the B^i's ensures that the sets induced by the independent sequence (as in item 1 of the definition) are indeed minterms of $\mathcal{A}$, rather than just $A^i \cup B^i$. This is achieved by eliminating any constraints from other values of i. Now, as B is sparse, this value $2^i - 1$ is only slightly smaller as a function of the last party index in A^i, leading to a bound of $2^{i-g(i)} - 1$ for an arbitrarily small function $g = \omega(1)$, controlled by the 'sparsity' of the set B. This bound holds for all values of i. In our choice of $\mathcal{A}_f$, we also apply Theorem 1.6 infinitely many times to sets (A^i, B^i), making sure that the sets induced by the independent sequence in $A^i \cup B^i$ with respect to A^i are minterms of the entire AS $\mathcal{A}_f$. One global constraint that is imposed on minterms of a partite function, is that they need to include parties from consecutive pairs $\{p_{j,0}, p_{j,1}\}$ in some interval $j \in [t]$, and exactly one from each pair. Thus, we use a different shape of (A^i, B^i) pairs. Each $(A^i = p_{1,1}, \ldots, p_{\log t_i}, p_{\log t_i+1,0}, p_{\log t_i+1,1}, p_{t_i,0}, p_{t_i,1}, B^i = \{p_{t_i+1,0}, \ldots, p_{t_i+2^{t_i},0}\})$ are consecutively located sets of parties. The independent sequence consists of all subsets of A^i consisting of all the "unpaired" $p_{j,1}$'s at the beginning (referred next as the "1's prefix"), and some combination of $p_{j,b}$'s from all the available party pairs - $2^{t_i - \log t_i}$ in total. The t_i's is a fast-growing sequence (we picked a double exponential growth). The reason is that we want to maintain the $p_{j,1}$'s prefix part in the A^i's relatively small to $|A^i|$, as it decreases the length of the independent sequence for A^i. The reason we keep the 1's prefix at all is to make sure no subset of it was previously set to be a minterm (because the B^i-part only includes $p_{j,0}$'s). The bound on total share complexity for each A^i is $2^{t_i - \log t_i}$. We could make it arbitrarily close to 2^{t_i} by picking an even faster growth rate of the t_i's. On the other hand, it makes the density t_i's for which the total share size of parties in $[t_i]$ is large ($2^{t_i/2 - o(t_i)}$) decreases as the t_i sequence grows faster. This is not a problem for us, as all we need to prove Theorem 1.5 is an infinite such sequence, no matter how sparse.

This bound immediately implies a lower bound of $2^{t-o(t)}$ on total message complexity on any evolving CDS for f corresponding to $\mathcal{A}_f$ via Theorem 1.2. Finally, we point out that our lower bound on evolving CDS complexity is tight, as a monotone variant of the simple tree-based generic construction of [24] results in a $2^t - 1$ upper bound on the message complexity of evolving CDS for any evolving predicate.

Remark 1.7. We do not rely on the so called Infinite independent sequences as defined in [1] that involve a pair of party subsets $A, B \subseteq P$, which in turn generalizes the hard example by [28]. They obtain both (nearly) tight lower bound and the bound applies to all sufficiently large prefixes $[i]$ of the party set. As discussed above, this structure of (A^i, B^i)'s does not apply to our case due to the restricted form of minterms of partite functions.

2 Preliminaries

We denote vectors by bold small letters, and functions by plain small English letters. For a vector $\mathbf{u}$, we denote its i'th element by $\mathbf{u}_i$ or by $\mathbf{u}[i]$. . For an integer $t \in \mathbb{N}$, $[t]$ denotes the set $\{1 \leq i \leq t | i \in \mathbb{N}\}$. For a set A, its power set is denoted by 2^A.

2.1 Conditional Disclosure of Secrets (CDS)

We present the definition of a k-server conditional disclosure of secrets protocols that defined by [21].

Definition 2.1 (Conditional Disclosure of Secrets Protocol). Let $f : X_1 \times \cdots \times X_k \to \{0,1\}$ be a k-input function. A k-server CDS protocol $\mathcal{P}$ for f with domain of secrets s consists of:

- A finite domain of common random strings R, and k finite message domains $M_1, \ldots, M_k$
- Deterministic message computation functions $\mathrm{ENC}_1, \ldots, \mathrm{ENC}_k$, where $\mathrm{ENC}_i : X_i \times S \times R \to M_i$ for every $i \in [k]$ (we also say that $\mathrm{ENC}_i(x_i, s, r)$ is the message sent by the i-th server to the referee).
- A deterministic reconstruction function $\mathrm{DEC} : X_1 \times \cdots \times X_k \times M_1 \times \cdots \times M_k \to \{0,1\}$.

We denote $\mathrm{ENC}(x, s, r) = (\mathrm{ENC}_1(x_1, s, r), \ldots, \mathrm{ENC}_k(x_k, s, r))$. We say that a CDS protocol $\mathcal{P}$ is a CDS protocol for a function f if the following two requirements hold:

Correctness. For any input $(x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$ for which $f(x_1, \ldots, x_k) = 1$, every secret $s \in S$, and every common random string $r \in R$,

$$\mathrm{DEC}(x_1, \ldots, x_k, \mathrm{ENC}_1(x_1, s, r), \ldots, \mathrm{ENC}_k(x_k, s, r)) = s.$$

Privacy. For any input $x = (x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$ for which $f(x_1, \ldots, x_k) = 0$ and for every pair of secrets s_1, s_2, the distributions $\mathrm{ENC}(x, s_1, r)$ and $\mathrm{ENC}(x, s_2, r)$ are identical, where the distributions are over the choice of r from R at random with uniform distribution.

The secret size is defined as $\log |S|$. The message size of a CDS protocol $\mathcal{P}$ is defined as the size of largest message sent by the servers, i.e., $mc_{\mathcal{P},\max} = \max_{1 \leq i \leq k}\{\log |M_i|\}$, and the total message size is defined as $mc_{\mathcal{P}} = \sum_{i=1}^{n} \log |Mi|$.

We use the following, state of the art general CDS protocol.

Theorem 2.1 ([26]). *For any k-input functions $f : [n]^k \to \{0,1\}$ there is a $k-$server CDS protocol with a one bit secret and message size $n^{O(\sqrt{k/\log n}\log(k\log n))}$. In particular, for $X_i = \{0,1\}$ $(n = 2)$, the protocol has message complexity $2^{O(k^{0.5}\log k)}$.*

We adopt the following notions of evolving CDS functions and protocols from [30].

Definition 2.2 (**Evolving Predicates** [30]). Let $\{X_t\}_{t\in\mathbb{N}}$ denote a sequence of finite input domains. An evolving predicate is a function $f : \cup_{t\in\mathbb{N}} X_1 \times \ldots \times X_t \to \{0,1\}$. For every t We denote by $f_t : X_1 \times \ldots \times X_t \to \{0,1\}$ the predicate f restricted to $X_1 \times \ldots \times X_t$ (that is, $f_t(\mathbf{x}) = f(\mathbf{x})$). We require that f is monotone in the sense that for every $t \in \mathbb{N}$ and every $(x_1, \ldots, x_{t+1}) \in X^{t+1}$, it holds that

$$f_{t+1}(x_1, \ldots, x_{t+1}) \geq f_t(x_1, \ldots, x_t).$$

Note that Definition 2.2 implies that if f_t holds on the input $(x_1, \ldots, x_t) \in X_1 \times \cdots \times X_t$, then f_i also holds on $(x_1, \ldots, x_t, x_{t+1}, \ldots, x_i)$ for every $(x_{t+1}, \ldots, x_i) \in X_{t+1} \times \ldots \times X_i$, and if f_t does not hold on the input $(x_1, \ldots, x_t) \in X_1 \times \cdots \times X_t$, then f_i also does not hold on $(x_1, \ldots, x_i)$ for every $i \in [t-1]$.

Definition 2.3 (**Evolving Conditional Disclosure of Secrets Protocols** [30]). Let $\{X_t\}_{t\in\mathbb{N}}$ be a sequence of input domains, and let $f : \cup_{t\in\mathbb{N}} X_1 \times \ldots \times X_t \to \{0,1\}$ be an evolving predicate. An evolving conditional disclosure of secrets protocol $\mathcal{P}$ with domain of secrets S, sequence of domains of random strings $\{R_t\}_{t\in\mathbb{N}}$, and sequence of message domains $\{M_t\}_{t\in\mathbb{N}}$, consists of a sequence of deterministic message computation algorithms $\{\mathrm{ENC}_t : X_t \times S \times R_1 \times \cdots \times R_t \to M_t\}_{t\in\mathbb{N}}$, that takes an input, a secret, and t random strings, and outputs a message for party Q_t, and a sequence of deterministic reconstruction algorithms $\mathrm{DEC}_t : X_1 \times \cdots \times X_t \times M_1 \times \cdots \times M_t \to S$, for every $t \in \mathbb{N}$, that takes t inputs and t messages, and outputs a secret. We say that $\mathcal{P}$ is an evolving CDS protocol for the evolving predicate f if for every $t \in \mathbb{N}$, the CDS protocol $\mathcal{P}_t$, with the message computation algorithms $\mathrm{ENC}_t^i : X_i \times S \times R_1 \times \cdots \times R_t \to M_i$, for every $i \in [t]$, where $\mathrm{ENC}_i^t(x_i, s, r_1, \ldots, r_t) := \mathrm{ENC}_i(x_i, s, r_1, \ldots, r_i)$, and the reconstruction algorithm DEC_t, is a CDS protocol for the predicate f_t, as in Definition 2.1.

The total (maximal) message size of an evolving CDS protocol $\mathcal{P}$ is defined for every t, as the message complexity of the prefix $[t]$ of servers (q_t). We denote by $emc_{\mathcal{P},\max}(t) = mc_{\mathcal{P}_t,\max}$ the message size of server q_t, and the total message size is defined as $emc_{\mathcal{P}}(t) = mc_{\mathcal{P}_t}$. We say $ems(t)$ is a maximal (total) message complexity bound for an evolving predicate f, if for every evolving CDS protocol $\mathcal{P}$ implementing f, and for every t, $emc_f(t) \geq emc_{\mathcal{P}}(t)$ $(emc_f(t) \geq mc_{\mathcal{P}_t,\max})$. In the following, we refer to maximal message complexity as just *message complexity*, inspired by the default notion of share complexity in an evolving secret-sharing scheme.

2.2 Secret-Sharing Schemes

We start by defining (perfect) secret-sharing schemes for a finite set of parties.

Definition 2.4 (Access Structures). Let $P = \{p_1, \ldots, p_n\}$ be a set of parties. A collection $\Gamma \subseteq 2^{\{p_1,\ldots,p_n\}}$ is monotone if $B \in \Gamma$ and $B \subseteq C$ imply that $C \in \Gamma$. An access structure $\Gamma \subseteq 2^{\{p_1,\ldots,p_n\}}$ is a monotone collection of non-empty sets. Sets in Γ are called authorized, and sets not in Γ are called unauthorized. We will represent an n-party access structure by a function $f : \{0,1\}^n \to \{0,1\}$, where an input (i.e., a string) $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n \in \{0,1\}^n$ represents the set $A_\sigma = \{p_i : i \in [n], \sigma_i = 1\}$, and $f(\sigma) = 1$ if and only if $A \in \mathcal{A}$. We will also call f an access structure.

A secret-sharing scheme defines a way to distribute shares to parties. Such a scheme is said to realize an access structure Γ if the shares held by any authorized set of parties (i.e., a set in the access structure) can be used to reconstruct the secret, and the shares held by any unauthorized set of parties reveal nothing about the secret. The formal definition is given as follows.

Definition 2.5 (Secret-Sharing Schemes)**.** A secret-sharing scheme Π over a set of parties $P = \{p_1, \ldots, p_n\}$ with domain of secrets S and domain of random strings R is a mapping from $S \times R$ to a set of n-tuples $S_1 \times S_2 \times \cdots \times S_n$ (the set S_j is called the domain of shares of p_j). A dealer distributes a secret $s \in S$ according to Π by first sampling a random string $r \in R$ with uniform distribution, computing a vector of shares $\Pi(s; r) = (\mathbf{Sh}_1, \ldots, \mathbf{Sh}_n)$, and privately communicating each share $\mathbf{Sh}_j$ to party p_j. For a set $A \subseteq \{p_1, \ldots, p_n\}$, we denote $\Pi_A(s; r)$ as the restriction of $\Pi(s; r)$ to its A-entries (i.e., the shares of the parties in A).

A secret-sharing scheme Π with domain of secrets S realizes an access structure Γ if the following two requirements hold:

Correctness. The secret s can be reconstructed by any authorized set of parties. That is, for any authorized set $B = \{p_{i_1}, \ldots, p_{i_{|B|}}\} \in \Gamma$, there exists a reconstruction function $\mathrm{Recon}_B : S_{i_1} \times \cdots \times S_{i_{|B|}} \to S$ such that for every secret $s \in S$ and every random string $r \in R$, it holds that $\mathrm{Recon}_B\left(\Pi_B(s; r)\right) = s$.

Security. Every unauthorized set cannot learn anything about the secret from its shares. Formally, for any set $T \notin \Gamma$, every two secrets $s_1, s_2 \in S$, and every possible vector of shares $\langle \mathbf{Sh}_j \rangle_{p_j \in T}$, $\Pr\left[\Pi_T(s_1; r) = \langle \mathbf{Sh}_j \rangle_{p_j \in T}\right] = \Pr\left[\Pi_T(s_2; r) = \langle \mathbf{Sh}_j \rangle_{p_j \in T}\right]$, where the probability is over the choice of r from R with uniform distribution.

The size of the share of party p_j is defined as $\log |S_j|$ and the size of the shares of Π is defined as $sc_{\Pi,\max} = \max_{1 \le j \le n} \log |S_j|$. The total share size of Π is defined as $sc_{\Pi} = \sum_{j=1}^{n} \log |S_j|$.

In an evolving secret-sharing scheme, defined by [24], the number of parties is not known in advanced and could potentially be infinite. Parties arrive one after the other; when a party p_t arrives the dealer gives a share only to him. The dealer cannot update the share later and does not know how many parties will arrive after party p_t. Thus, we measure the share size of p_t as a function of t. We defining an evolving access structure, which specifies the authorized sets. The number of parties in an evolving access structure is infinite, however every authorized set is finite.

Definition 2.6 (Evolving Access Structures)**.** Let $P = \{p_i\}_{i \in \mathbb{N}}$ be an infinite set of parties. A collection of finite sets $\Gamma \subseteq 2^P$ is an *evolving access structure* if for every $t \in \mathbb{N}$ the collections $\Gamma^t \triangleq \Gamma \cap 2^{\{p_1, \ldots, p_t\}}$ is an access structure as defined in Definition 2.4. We will represent an access structure by a function $f : \{0,1\}^* \to \{0,1\}$, where an input (i.e., a string) $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n \in \{0,1\}^n$ represents the set $A_\sigma = \{p_i : i \in [n], \sigma_i = 1\}$,[4] and $f(\sigma) = 1$ if and only if $A_\sigma \in \Gamma$. We will also call f an evolving access structure.

Definition 2.7 (Evolving Secret-Sharing Schemes)**.** Let S be a domain of secrets, where $|S| \ge 2$, and $\{R_t\}_{t \in \mathbb{N}}, \{S_t\}_{t \in \mathbb{N}}$ be two sequences of finite sets. An *evolving secret-sharing scheme* with domain of secrets S is a sequence of mappings $\Pi = \left\{\Pi^t\right\}_{t \in \mathbb{N}}$, where for every $t \in \mathbb{N}$, Π^t is a mapping $\Pi^t : S \times R_1 \times \cdots \times R_t \to S_t$ (this mapping returns the share $\mathbf{Sh}_t$ of p_t).

[4] In particular, the same set has infinitely many representations by inputs of various lengths, using sufficiently many trailing zeros.

An evolving secret-sharing scheme $\Pi = \left\{\Pi^t\right\}_{t\in\mathbb{N}}$ realizes an evolving access structure Γ if for every $t \in \mathbb{N}$ the secret-sharing scheme $\Pi_t\left(s;\left(r_1,\ldots,r_t\right)\right) \triangleq \left\langle \Pi^1\left(s;r_1\right),\ldots,\Pi^t\left(s;r_1,\ldots,r_t\right)\right\rangle$ (i.e., the shares of the first t parties) is a secret-sharing scheme realizing Γ^t according to Definition 2.5.

2.2.1 On monotonicity

In the following section, we compare the monotonicity of an evolving access structure of an evolving secret-sharing scheme with the monotonicity of evolving predicate of evolving CDS protocol. We use vectors (consisting of 0s and 1s) to represent inputs.

In evolving predicate, the monotonicity requirement is:

1. For every $t \in \mathbb{N}$ and every $(x_1,\ldots,x_{t+1}) \in X^{t+1}$, it holds that,

$$f_{t+1}(x_1,\ldots,x_{t+1}) \geq f_t(x_1,\ldots,x_t)$$

   (i.e., if we extend a vector, the function's value on it cannot decrease).

In an evolving access structure, we require the monotonicity condition of evolving predicate but add additional requirements:

1. For every $t \in \mathbb{N}$, every $i \in \mathbb{N}$ and every $(x_1,\ldots,x_t) \in X^t$, it holds that,

$$f_t(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_t) \geq f_t(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_t)$$

   (i.e., if we change a 0 to a 1 in a vector, the function's value on that vector cannot decrease).

2. For every $t \in \mathbb{N}$ and every $(x_1,\ldots,x_{t+1}) \in X^{t+1}$, it holds that,

$$f_{t+1}(x_1,\ldots,x_t,0) = f_t(x_1,\ldots,x_t)$$

   (i.e., extending a vector by any number of zeros at the end does not change the function's value on the vector).

3. From 2, it follows that every input has an infinite representation of vectors.

In conclusion, in an evolving secret-sharing scheme, monotonicity is "stronger," and it is more convenient to view inputs as a set of parties rather than as vectors, due to the multiple representations of vectors. In an evolving CDS protocol, monotonicity is "weaker," and we think of inputs as vectors where their length is important.

In the paper, when we want to align the properties of different objects with an evolving CDS protocol, we refer to them as "non-monotone" because they do not satisfy the "strong" monotonicity of an evolving secret-sharing scheme.

2.3 Infinite Decision Trees

Now, we present the evolving secret-sharing scheme for infinite decision trees (IDT), as formalized in [1], with the modification that the IDT does not include edges labeled with 1 (as implied in [24, 25]).

Definition 2.8 (**Infinite decision trees – IDT** [1])**.** An infinite decision tree $T = (G = (V, E), u_0 = 0, \mu)$ is a special case of GIDT, where each edge (u, v) is labeled by a variable x_v, where for simplicity we assume that $V = \mathbb{N} \cup \{0\}$ (i.e., a vertex is a non-negative integer). As G is a tree, each variable labels at most one edge. Furthermore, we assume that the vertices are ordered by the layers, i.e., $L_0 = \{0\}$, $L_1 = \{1, \ldots, w(1)\}$, and so on (where $w(i)$ is the width of layer L_i). Thus, the variables on edges entering layer L_i are from the set $\{x_j : j \in L_i\}$. An IDT is a computational model which defines monotone evolving AS's (equivalently, monotone functions $f : \{0,1\}^* \to \{0,1\}$) in a natural way. That is, the function induced by an IDT T, $f(x_1, \ldots, x_t) = 1$, if there exists a path from the root to a leaf where all edges are satisfied by $x_1, \ldots, x_t$.

Construction 2.2 (An evolving secret-sharing scheme Π_{IDT} for an IDT $T = (G, u_0, \mu)$ [1])**.**
Input: $s \in \{0, 1\}$.
The sharing algorithm:

- For $i = 1$ to ∞:
  - For every vertex $u \in L_{i-1}$ and $v \in L_i$, when party p_v arrives choose a bit r_v as follows:
    - If v is a leaf, then let $u_0, v_1, \ldots, v_{t-1}, v$ be the path from the root u_0 to v in G and assign $r_v \leftarrow s \oplus \bigoplus_{j=1}^{t-1} r_{v_j}$.
    - If v is not a leaf and $\mu_{(u,v)} = x_v$, then r_v is a uniformly distributed random bit.
  - The share of p_v is $\mathbf{Sh}_v = r_v$.

Claim 2.3 ([1])**.** The evolving secret-sharing scheme Π_{IDT} realizes the infinite decision tree $T = (G, u_0, \mu)$, where the share of every p_t is a bit.

3 Additional notions of CDS

We will need a notion of a strong CDS which handles cases where there are servers that fail to arrive. This case is clear cut in the standard CDS setting - we require all servers to arrive and contribute an input. Thus, if at least one of them does not, the secret should not be revealed - see Construction 3.2 below. However, we obviously can not require that all infinitely many servers arrive in the evolving CDS. The set of servers contributing inputs should be some (finite) prefix $[n]$ of $\mathbb{N}$, as implied by the definition of the range of evolving predicates. Thus, it makes sense to require that the secret is learned only if $f(\mathbf{x}[1, t]) = 1$ for some prefix of the set of available inputs $\mathbf{x}_A$. As we did not find an explicit definition of this notion, we define both the standard and evolving variants, dubbing it *strong* CDS.

Definition 3.1 (**Strong Conditional Disclosure of Secrets Protocol**)**.** Let $f : X_1 \times \cdots \times X_k \to \{0, 1\}$ be a function. A k-server strong CDS protocol $\mathcal{P}$ for f with domain of secrets s, randomness domain R and message domains $M_1, \ldots, M_k$ is a CDS protocol for it as in Definition 2.1, with the following additional privacy requirement. Let us denote,

$$\text{ENC}^{-i}(x, s, r) = (\text{ENC}_1(x_1, s, r), \ldots, \text{ENC}_{i-1}(x_{i-1}, s, r), \text{ENC}_{i+1}(x_{i+1}, s, r), \ldots \text{ENC}_k(x_k, s, r)).$$

For any input $x = (x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k$, for every $i \in [k]$ and for every pair of secrets s_1, s_2, the distributions $\text{ENC}_{[k]\setminus i}(x, s_1, r)$ and $\text{ENC}_{[k]\setminus i}(x, s_2, r)$ are identical, where r is sampled uniformly from R.

Strong standard CDS can be easily obtained from standard CDS by a simple transformation, with almost no overhead. Essentially, it boils down to sharing a masked secret via the standard scheme, and appending an additive share of the mask to every CDS share. We include it here for completeness.

Construction 3.1 (Strong CDS Protocol Π'(folklore))**.** 1. Let $\Pi = (\mathrm{ENC}, \mathrm{DEC})$ denote a k-server CDS protocol for a function $f : X_1 \times \ldots \times X_k \to \{0,1\}$ and secret domain S. A strong CDS protocol Π' over S for f is as follows.

2. $\mathrm{ENC}'_i(x_i, s; r' = (r, b))$: Output $(\mathrm{ENC}_i(x_i, s + b(\ \mathrm{mod}\ ; |S|), r), sh_i)$, where sh_i is the i'th share in an additive sharing of b.

3. $\mathrm{DEC}'(\mathbf{x}, (v_{1,1}, v_{1,2}), \ldots, (v_{k,1}, v_{k,2}))$: 1. Recover $s' = s + b$ via $\mathrm{DEC}(\mathbf{x}, v_{1,1}, \ldots, v_{k,1})$, then recover $b = \sum_{j \in [v]} v_{k,2}(\ \mathrm{mod}\ ; |S|)$, and output $s' - b(\ \mathrm{mod}\ ; |S|)$.

For the evolving case we do not have a general black box transformation, but we do devise a simple transformation from the evolving CDS we construct into a strong evolving CDS (in the text, for simplicity we directly construct a strong evolving CDS). It is an interesting open question to devise a general trnasformation from evolving CDS to strong evolving CDS, with a small overhead in share complexity.

Definition 3.2 (**Strong Evolving Conditional Disclosure of Secrets Protocol**)**.** Let f be an evolving predicate specified by $f_t : X_1 \times \cdots \times X_t \to \{0,1\}_{t \in \mathbb{N}}$. An evolving conditional disclosure of secrets protocol $\mathcal{P}$ with domain of secrets S, sequence of domains of random strings $\{R_t\}_{t \in \mathbb{N}}$, and sequence of message domains $\{M_t\}_{t \in \mathbb{N}}$ is said to be a strong evolving CDS implementing f if. (1) It is an evolving CDS implementing f with parameters as above. (2) For every $x_1, \ldots, x_t$, for every $i \leq t$ and for every pair of secrets $s_1, s_2 \in S$, if $F(x_1, \ldots, x_{i-1}) = 0$ the distributions $\mathrm{ENC}^{-i}(x_t, s_1, r = (r_1, \ldots, r_t))$ and $\mathrm{ENC}^{-i}(x_t, s_2, r = (r_1, \ldots, r_t))$ are identical, where r is uniformly sampled from $R_1 \times \ldots \times R_t$.[5]

Let us point out a certain subtlety of the definition of strong evolving CDS.

Remark 3.2. The standard definition of evolving CDS already involves a certain degree of 'strength' in the sense that for an input $x = (x_1, \ldots, x_t)$, we receive CDS messages for each of its t prefixes, using the corresponding $\mathcal{P}_t$ CDS protocols with correlated randomness. We would like no information to leak about the secret in case f evaluates to 0 on all of these prefixes. The privacy guarantees we require, however, are just for each $\mathcal{P}_t$, for its corresponding f_t predicates. It is not clear in advance why the privacy of each $\mathcal{P}_i$ for its own input domain implies that we do not learn anything from $\mathcal{P}_j$'s message for $(x_1 \ldots, x_j)$ combined with messages for all inputs $(x_1, \ldots, x_j, \ldots, x_i)$ for $j < i < t$ that use correlated randomness for their own inputs. It does work, since $f(x_1, \ldots, x_t) = 0$, and f is monotone, so s is not implied by the messages in $\mathcal{P}_i$, a subset of which corresponds to the messages in $\mathcal{P}_j$, thus no leakage occurs. Now, when the message corresponding to index $i \in [t]$ is missing, we would like that the secret is learned iff. $f(x_1, \ldots, x_{i-1}) = 1$ (that is, f is 1 on the last contiguous prefix of $\mathbf{x}$). But, it could be that $f(x_1, \ldots, x_{i-1}, v, x_{i+1}, \ldots, x_t) = 1$ for all $v \in X_t$. Then, the above argument does no longer hold, and we need to make an explicit requirement of "strength" for this case. We would automatically get strength for $\mathcal{P}$ if all protocols $\mathcal{P}_i$ induced by $\mathcal{P}$ were strong (finite) CDS protocols for their corresponding f_i's. Unlike the finite case, it is not clear how to

[5] Viewing $(\mathrm{ENC}_1, \ldots, \mathrm{ENC}_t)$ as a t-party standard CDS with randomness domain $R = R_1 \times \cdots \times R_t$.

achieve this, since applying the naive transformation from Construction 3.1 would result in infinite message sizes.

4 Strong Evolving CDS Protocols for Infinite Non-Monotone Branching Programs

In [1], an evolving secret-sharing scheme based on infinite non-deterministic branching programs (IBP) was constructed. In a nutshell, the construction relied on converting the IBP into a so called (monotone) generalized infinite tree (GIDT), based on which the evolving secret-sharing scheme was constructed (formalizing the technique of [24, 25]). In short, the construction proceeds on two levels. It first performs a variant of an evolving secret-sharing scheme for undirected connectivity for the GIDT, which assigns 'virtual' shares to every edge, and then shares each of the virtual shares via a finite scheme for the predicate of each share (thus the 'generalized', rather than predicates depending on a single variable as in IDT), among the parties on which that edge depends.

In this section, we define infinite non-monotone non-deterministic branching programs, and use an approach similar to the above to construct evolving CDS protocols. It turns out that replacing every edge's share by the appropriate strong CDS shares results in a strong evolving CDS for f. We need the strong variant of evolving CDS for our subsequent improved CDS-based evolving secret-sharing schemes construction. As it does not cost almost any overhead in share size of conceptual complexity, we prefer to directly construct a strong variant, without a regular one (which could be obtained by using standard CDS for the edges' predicates).

An infinite non-monotone non-deterministic branching program (INBP) is a generalization of infinite monotone non-deterministic branching programs that defined in [1], except that the edge can label non-monotone variable $\neg x_i$.

Definition 4.1 (Infinite Non-Monotone Non-Deterministic Branching Programs–INBP). An infinite non-monotone non-deterministic branching program is a triple $B = (G = (V, E), u_0, \mu)$, where V is a countable set of vertices, G is an infinite directed acyclic graph, u_0 is a source vertex, and $\mu : E \to \{x_i : i \in \mathbb{N}\} \cup \{\neg x_i : i \in \mathbb{N}\} \cup \{1\}$ is a labeling of the edges by variables or by 1 (we will sometimes use the notation μ_e instead of $\mu(e)$). We denote by U_{leaf} the set of vertices in V with out-degree 0, i.e., the leaves.

For a path P in the branching program and a finite input (an assignment for the variables on the path) $\sigma \in \{0,1\}^t$ for some $t \in \mathbb{N}$, we say that σ satisfies P and denote $\text{sat}_P(\sigma) = 1$ if σ satisfies all variables on the path, i.e., for each edge e on the path either $\mu_e = 1$ or $\mu_e = x_i$ for some $1 \leq i \leq t$ such that $\sigma_i = 1$ or $\mu_e = \neg x_i$ for some $1 \leq i \leq t$ such that $\sigma_i = 0$. And in particular, no μ_e on P depends on a variable x_i for $i > t$ (otherwise we say $\text{sat}_P(\sigma) = 0$).[6] The branching program accepts an input σ if there exists a directed path P starting in the source vertex u_0 and leading to some leaf $u \in U_{\text{leaf}}$ such that $\text{sat}_P(\sigma) = 1$. The function $f : \{0,1\}^* \to \{0,1\}$ computed by B is the function f such that $f(\sigma) = 1$ if and only if B accepts σ.

A LINBP is a private case of INBP, defined over layered (infinite, directed, acyclic) graphs (i.e., with edges going only from any one layer to its consecutive layer) with the additional requirement that the label of any edge from layer $i-1$ to layer i is either x_i or $\neg x_i$ or 1.

[6] Note that for IBP there was no such issue, as it handled only monotone functions in the sense that complementing a vector x with infinitely many 0's could not flip the value of f to 1, so we could always assume this. In the CDS setting there is no such asymmetry, so we must handle the case of inputs that are actually missing.

Definition 4.2 (**Layered INBP–LINBP**). An infinite non-monotone non-deterministic branching program is *layered* if the vertices of G can be partitioned into finite sets $(L_i)_{i\in\mathbb{N}\cup\{0\}}$, such that $L_0 = \{u_0\}$, there are edges only from layer $i-1$ to layer i for some $i \in \mathbb{N}$, and all edges entering layer i are labeled either by x_i or $\neg x_i$ or 1. For a vertex $u \in V$, we denote $L(u)$ as the layer of u, i.e., the index i such that $u \in L_i$. The width of the branching program at layer i, denoted $w(i)$, is the number of vertices in layer L_i. For a LINBP $B = (G, u_0, \mu)$ and vertices u, v, we define the predicate $\text{reach}_{u,v}$ as $\text{reach}_{u,v} = 0$ if there is no path in G from u to v, and otherwise

$$\text{reach}_{u,v}(x_{L(u)+1}, \ldots, x_{L(v)}) = \bigvee_{\substack{P \text{ is a path in } G \\ \text{from } u \text{ to } v}} \text{sat}_P(x_{L(u)+1}, \ldots, x_{L(v)}),$$

i.e., an input satisfies $\text{reach}_{u,v}$ if and only if it satisfies at least one path from u to v. We stress that, unlike Definition 4.1, here we consider an assignment to the predicate sat_P that only contains the variables that can appear on the path, i.e., we consider assignments $\sigma_{L(u)+1}, \ldots, \sigma_{L(v)}$ to the variables $x_{L(u)+1}, \ldots, x_{L(v)}$.

We modify [1]'s definition to define a generalized infinite decision tree that also handles non-monotone predicates. We define a generalized infinite non-monotone decision tree, is an infinite tree graph such that each edge, instead of being labeled by a variable x_i, is labeled by a predicate of some variables $x_i, \ldots, x_j$. The variables are divided into generations $\{G_i\}_{i\in\mathbb{N}}$, and an edge of distance i from the root be labeled with a predicate on the variables in generation G_i.

Definition 4.3 (**Generalized Infinite Non-Monotone Decision Trees – GINDT**). A generalized infinite non-monotone decision tree is a quadruple $T = (G = (V, E), u_0, \mu, h)$, where

- V is a countable set of vertices,
- $G = (V, E)$ is an infinite directed tree with root vertex u_0 such that the out-degree of each vertex is finite. We denote the i^{th} level L_i as $\{u \in V : u \text{ is at distance } i \text{ from } u_0\}$, and refer to L_i as the i^{th} *layer*.
- $h : \mathbb{N} \to \mathbb{N}$ is an increasing function that partitions the variables into generations, where for $i \in \mathbb{N}$, generation i is the variable set $G_i \triangleq \{x_{h(i-1)+1}, \ldots, x_{h(i)}\}$ (where we define $h(0) = 0$),
- μ is a labeling of the edges by *predicates*, where for every edge $e = (u, v)$ where $u \in L_{i-1}$ to level $v \in L_i$, the labeling μ_e is any predicate on the variables of generation i, of the form $\varphi_e(x_{h(i-1)+1}, \ldots, x_{h(i)}) : \{0,1\}^{h(i)-h(i-1)} \to \{0,1\}$. If v is a leaf, $\varphi_e(x_{h(i-1)+1}, \ldots, x_j) : \{0,1\}^{j-h(i-1)} \to \{0,1\}$ may (syntactically) depend on a prefix $(x_{h(i-1)+1}, \ldots, x_j)$ of $x_{h(i-1)+1}, \ldots, x_{h(i)}$.

For a path P in the tree ending at a vertex in level i, we say that P is satisfied by an input $\sigma \in \{0,1\}^t$, denoted by $\text{sat}_P(\sigma) = 1$, if $h(i) \leq t$ (that is, the variables in all predicates labeling edges in P are from $x_1, \ldots, x_t$) and for each edge e on the path the predicate μ_e is satisfied by σ. The GINDT T accepts an input σ if there is at least one directed path P starting in the source vertex u_0 and leading to a leaf such that $\text{sat}_P(\sigma) = 1$. The function $f : \{0,1\}^* \to \{0,1\}$ computed by T is the function f such that $f(\sigma) = 1$ if and only if T accepts σ.

We next show how to realize an evolving predicate f of a GINDT using the secret-sharing scheme Π_{IDT} realizing a related infinite decision tree (where edges are labeled by variables). In a GINDT each edge e is labeled by a predicate μ; in the following protocol $\mathcal{P}_{\text{GINDT}}$ we consider this predicate as describing an evolving predicate f over the servers of the generation.

Construction 4.1 (An Evolving CDS protocol $\mathcal{P}_{\text{GINDT}}$ for a GINDT $T = (G = (V, E), u_0, \mu, h)$)**.**
Input: $s \in \{0, 1\}$.

- Construct from the GINDT $T = (G = (V, E), u_0, \mu, h)$ an IDT $T' = (G = (V, E), u_0, \mu')$ whose variables are $\{y_i : i \in \mathbb{N}\} \cup \{\neg y_i : i \in \mathbb{N}\}$, where for every edge $(u, v) \in E$ we have $\mu'(u, v) = y_v$.
- Execute the scheme Π_{IDT} for T' and use its shares as follows:

  ($*$ Recall that in Π_{IDT} the parties arrive according to layers, where inside a layer the order is some arbitrary fixed order $*$)
- For $i = 1$ to ∞ do:
  - When server $q_{h(i-1)+1}$ arrives do:
    - For every $(u, v) \in E$, where $u \in L_{i-1}$, $v \in L_i$, generate the share r_v of y_v in the scheme Π_{IDT} and use r_v as a secret of a strong CDS protocol that realizing the function defined by $\mu_{(u,v)}$ among the servers $q_{h(i-1)+1}, \ldots, q_{h(i)}$.
    - Let $\mathbf{MS}_t$, for $h(i-1)+1 \leq t \leq h(i)$, be the concatenation of the messages of q_t in all these protocols.
    - Give server $q_{h(i-1)+1}$ the message $\mathbf{MS}_{h(i-1)+1}$.
  - For $t = h(i-1)+2$ to $h(i)$ do:
    - When server q_t arrives give it the massage $\mathbf{MS}_t$.

Claim 4.2. $\mathcal{P}_{\text{GINDT}}$ is a strong evolving CDS realizing the GINDT $T = (G, u_0, \mu, h)$'s function. For a server q_t in generation i (that is, $h(i-1)+1 \leq t \leq h(i)$), the size of the message of q_t is the sum of the sizes of the messages of q_t in the strong CDS protocols for $\mu_{(u,v)}$ for every $(u, v) \in E$ such that $u \in L_{i-1}$, $v \in L_i$ (there are at most $w(i)$ such protocols, where $w(i)$ is the number of vertices in tree layer i).

Proof. First we prove the correctness of the protocol. Let $\sigma^t = (\sigma_1, \ldots, \sigma_t)$ be an input of the first t servers accepted by T such that $f(\sigma^t) = 1$. Thus, there exists an accepting path P from u_0 to a leaf such that $\text{sat}_P(\sigma) = 1$. For every edge $e = (u, v) \in P$, the input σ satisfies μ_e. Thus, the t servers can reconstruct the Π_{IDT} share r_v by correctness of the strong CDS protocol realizing μ_e. By correctness of Construction 2.2, the t servers can reconstruct s.

Next we prove the security of the scheme. Let $\sigma^{t'}(\sigma_1, \ldots, \sigma_{t'}, \sigma_{t'+2}, \ldots, \sigma_t)$ be an input of the first t' servers with $f(\sigma^{t'}) = 0$, followed by a possibly empty input $\sigma^+ = (\sigma_{t'+2}, \ldots, \sigma_t)$ held by servers $t'+2, \ldots, t$ respectively. We prove that the $\mathcal{P}_{\text{GINDT}}$ messages corresponding to these inputs reveal nothing about the secret. Assume for contradiction that they do, so by privacy of Π_{IDT}, there exists a path $P = (u_0, v_1, \ldots, v_k)$ in T', where v_k is a leaf for which the servers learn all Π_{IDT} shares of edges on P. Assume $x_{t'+1}$ belongs to generation j of T. There are two cases. In one case, (v_{k-1}, v_k) is labeled by variables $x_1, \ldots, x_{t''}$ for $t'' \leq t'$. Thus all edges on subpath $(u_0, \ldots, v_{t''})$ depend on the prefix $\sigma^{t''}$ of $\sigma^{t'}$. In particular, $f(\sigma^{t''}) = 0$ since $f(\sigma^{t'}) = 0$ and f is monotone (in the sense of Item 1 of Section 2.2.1). Thus, $\text{sat}_P(\sigma^{t''}) = 0$, so there exists some $e \in P$ for which $\mu_e(\sigma^{t''}) = 0$. Thus, by privacy of the CDS used for μ_e (without using the strong property, even), the share for y_e in Π_{IDT} is not learned, contradicting the fact that all Π_{IDT} shares along P are learned. Otherwise, (v_{k-1}, v_k) is labeled by variables $x_1, \ldots, x_{t''}$ for $t'' \geq t'+1$. So, for the edge e

on the path P for which μ_e depends on $x_{t'+1}$, the Π_{IDT} share for y_e is not learned by privacy of *strong* CDS utilized for this edge.[7] This again is a contradiction to the fact that all Π_{IDT} shares along P are learned.

For the message size, server q_t obtains a message in the strong CDS protocol realizing $\mu_{(u,v)}$ for each edge (u, v) where $u \in L_{i-1}$ and $v \in L_i$. □

We next describe a transformation from a LINBP B to a GINDT T computing the same function.

We start with an informal description of the transformation. To transform an LINBP B to an IDT T (where each edge is labeled by a variable), we duplicate vertices and have in T a vertex $u_{0,j_1,\ldots,j_{i-1},j_i}$ for every path $u_0, u_{j_1}, \ldots, u_{j_{i-1}}, u_{j_i}$ in B starting from the root, and add an edge $(u_{0,j_1,\ldots,j_{i-1}}, u_{0,j_1,\ldots,j_{i-1},j_i})$ whose label is the label of the edge $(u_{j_{i-1}}, u_{j_i})$. The problem with this construction is that the resulting IDT is to big. To construct more efficient GINDT (which will result in more efficient evolving CDS protocls), we partition the variables into generations (described by a function $h : \mathbb{N} \to \mathbb{N}$), the vertices in layer i of T are $u_{0,j_1,j_2,\ldots,j_i}$ for vertices $u_0, u_{j_1}, u_{j_2}, \ldots, u_{j_i}$ in the layers $0, h(1), h(2), \ldots, h(i)$ in B respectively. That is, the number of vertices in the resulting GINDT is much smaller. Now an edge $(u_{0,j_1,\ldots,j_{i-1}}, u_{0,j_1,\ldots,j_{i-1},j_i})$ represents *all paths* in B from $u_{j_{i-1}}$ to u_{j_i}, i.e., the predicate of this edge is satisfied by an input σ if and only if σ satisfies some path in B from $u_{j_{i-1}}$ to u_{j_i}.

As explained in Section 2.2.1, in an evolving secret-sharing scheme like the evolving CDS protocol there is monotonicity in the sense that a qualified set always remains qualified no matter what is added to it later. But unlike the evolving secret-sharing scheme, in the evolving CDS protocol adding parties with label 0 can turn an unqualified set into a qualified one. Therefore, it is not possible to use the technique done in [1] when only one edge of a secret-sharing scheme was included all the leaves in the intermediate levels between $h(i)$ and $h(i+1)$ in LIBP. Now when we use predicate of CDS protocol, a separate edge must be created for each intermediate level in LINBP so that the length of the amount of parties is equal.The formal construction is described below.

Construction 4.3 (A Transformation from a LINBP to a GINDT)**.**
Input: A LIBP $B = (G = (V, E), u_0, \mu)$ and an increasing function $h : \mathbb{N} \to \mathbb{N}$. (∗ We use the following notation for the vertices of the LIBP B – the vertices in the i-layer of B are $L_i = \{u^i_1, \ldots, u^i_{w(i)}\}$ for $i \in \mathbb{N} \cup \{0\}$. ∗)
Output: A GIDT $T = (G' = (V', E'), u'_0, \mu', h)$.
The transformation:

- The vertices in layer i of the tree G' are $L'_0 = \{u_0\}$ and for $i \in \mathbb{N}$ define [8]

$$L'_i = \{u_{0,j_1,\ldots,j_i} : 1 \le j_1 \le w(h(1)), u^1_{j_1} \notin U_{\text{leaf}}, \ldots, 1 \le j_i \le w(h(i)), u^i_{j_i} \notin U_{\text{leaf}}\}$$
$$\bigcup \{v_{0,j_1,\ldots,j_{i-1},j} : 1 \le j_1 \le w(h(1)), u^1_{j_1} \notin U_{\text{leaf}}, \ldots, 1 \le j_{i-1} \le w(h(i-1)), u^{i-1}_{j_{i-1}} \notin U_{\text{leaf}},$$
$$h(i-1)+1 \le j \le h(i)\}.$$

  The vertices of G' are $V' = \cup_{i \in \mathbb{N} \cup \{0\}} L'_i$. The tree leaves are $U'_{\text{leaf}} = \cup_{i \in \mathbb{N}} \{v_{0,j_1,\ldots,j_{i-1},j} : 1 \le j_1 \le w(h(1)), u^1_{j_1} \notin U_{\text{leaf}}, \ldots, 1 \le j_{i-1} \le w(h(i-1)), u^{i-1}_{j_{i-1}} \notin U_{\text{leaf}}, h(i-1)+1 \le j \le h(i)\}$.

[7] Note that by definition of strong CDS, this holds even if $\mu_e = 1$.

[8] Note that the last index in $v_{0,\ldots,j_{i-1},j}$ corresponds to a layer index, rather than the index of a vertex inside a layer, as for the former indices.

- The edges are

$$E' = \left\{(u_{0,j_1,\ldots,j_{i-1}}, u_{0,j_1,\ldots,j_{i-1},j_i}) : i \in \mathbb{N}, u_{0,j_1,\ldots,j_{i-1},j_i} \in V'\right\} \bigcup \left\{(u_{0,j_1,\ldots,j_{i-1}}, v_{0,j_1,\ldots,j_{i-1},j}) : i \in \mathbb{N}, u_{0,j_1,\ldots,j_{i-1}}, v_{0,j_1,\ldots,j_{i-1},j} \in V'\right\}.$$

- For every $e = \left(u_{0,j_1,\ldots,j_{i-1}}, u_{0,j_1,\ldots,j_{i-1},j_i}\right) \in E'$, let $u = u_{j_{i-1}}^{h(i-1)}$, $v = u_{j_i}^{h(i)}$ and define

$$\mu'_e(x_{h(i-1)+1}, \ldots, x_{h(i)}) = \mathrm{reach}_{u,v}(x_{h(i-1)+1}, \ldots, x_{h(i)}).$$

- For every $e = \left(u_{0,j_1,\ldots,j_{i-1}}, v_{0,j_1,\ldots,j_{i-1},j}\right) \in E'$, let $u = u_{j_{i-1}}^{h(i-1)}$ and define

$$\mu'_e(x_{h(i-1)+1}, \ldots, x_{h(i)}) = \bigvee_{v \text{ is a leaf in layer } j \text{ in } B} \mathrm{reach}_{u,v}(x_{h(i-1)+1}, \ldots, x_j).$$

Claim 4.4. Construction 4.3 outputs a GINDT T which computes the same function as B. Furthermore, the number of vertices in layer i of T is $|L'_i| = \left(\prod_{1 \le j < i}(w(h(j))\right) \cdot (w(h(i)) + h(i) - h(i-1))$, where ct is the generation size of L_i.

Proof. We first prove the equivalence of B and T, that is, we prove that B accepts an input $\sigma = \sigma_1, \ldots, \sigma_t$ if and only if T accepts σ. Let ℓ be the generation of t, that is $h(\ell-1)+1 \le t \le h(\ell)$.

First assume that B accepts σ. W.l.o.g., assume that no proper prefix of σ is accepted by B (otherwise apply the following arguments to such minimal prefix). Then, there exists a path $P = (u_0^0, u_{j_1}^1, \ldots, u_{j_t}^t)$ in G where $u_{j_t}^t \in U_{\text{leaf}}$ and $\mathrm{sat}_P(\sigma) = 1$. Consider the path

$$P' = (u_0, u_{0,j_{h(1)}}, \ldots, u_{0,j_{h(1)},j_{h(2)},\ldots,j_{h(\ell-1)}}, v_{0,j_{h(1)},j_{h(2)},\ldots,j_{h(\ell-1)},t})$$

in G'. We partition the path P in G to sub-paths – for every $1 \le i \le \ell-1$, let $P^i = (u_{j_{h(i-1)}}^{h(i-1)}, \ldots, u_{j_{h(i)}}^{h(i)})$ and let $P^\ell = (u_{j_{h(\ell-1)}}^{h(\ell-1)}, \ldots, u_{j_t}^t)$. Since $\mathrm{sat}_P(\sigma) = 1$, we deduce that $\mathrm{sat}_{P^i}(\sigma_{h(i-1)+1}, \ldots, \sigma_{h(i)}) = 1$ for $1 \le i \le \ell - 1$ and $\mathrm{sat}_{P^\ell}(\sigma_{h(i-1)+1}, \ldots, \sigma_t) = 1$. By the definition of $\mathrm{reach}_{u,v}$, this implies that

$$\mathrm{reach}_{u_{j_{h(i-1)}}^{h(i-1)}, u_{j_{h(i)}}^{h(i)}}(\sigma_{h(i-1)+1}, \ldots, \sigma_{h(i)}) = 1$$

for every $1 \le i \le \ell - 1$ and $\mathrm{reach}_{u_{j_{h(\ell-1)}}^{h(\ell-1)}, u_{j_t}^t}(\sigma_{h(i-1)+1}, \ldots, \sigma_t) = 1$, where $u_{j_t}^t$ is a leaf in B. Thus, in T we have

$$\mathrm{sat}_{P'}(\sigma) = \left(\bigvee_{v \text{ is a leaf in layer } t \text{ in } B} \mathrm{reach}_{u_{j_{h(\ell-1)}}^{h(\ell-1)}, v}(\sigma_{h(\ell-1)+1}, \ldots, \sigma_t)\right) \wedge \left(\bigwedge_{1 \le i \le \ell-1} \mathrm{reach}_{u_{j_{h(i-1)}}^{h(i-1)}, u_{j_{h(i)}}^{h(i)}}(\sigma_{h(i-1)+1}, \ldots, \sigma_{h(i)})\right) = 1.$$

In the other direction, assume that T accepts σ. W.l.o.g., assume that no proper prefix of σ is accepted by T. Then in T there exists a path $P' = (u_0, u_{0,j_1}, \ldots, u_{0,j_1,\ldots,j_{\ell-1}}, v_{0,j_1,\ldots,j_{\ell-1},t})$ where $v_{0,j_1,\ldots,j_{\ell-1},t}$ is a leaf and $\mathrm{sat}_{P'}(\sigma) = 1$. This implies that for every $1 \le i \le \ell - 1$

$$\mu'_{(u_{0,j_1,\ldots,j_{i-1}}, u_{0,j_1,\ldots,j_{i-1},j_i})}(\sigma_{h(i-1)+1}, \ldots, \sigma_{h(i)}) = \mathrm{reach}_{u_{j_{i-1}}^{i-1}, u_{j_i}^i}(\sigma_{h(i-1)+1}, \ldots, \sigma_{h(i)}) = 1.$$

Thus, for each $1 \leq i \leq \ell-1$, there exists some path P^i from $u_{j_i}^{i-1}$ to $u_{j_i}^i$ in G such that $\mathrm{sat}_{P^i}(\sigma) = 1$. Furthermore, since $\mu'_{(u_{0,j_1,\ldots,j_{\ell-1}},v_{0,j_1,\ldots,j_{\ell-1},t})}(\sigma_{h(\ell-1)+1},\ldots,\sigma_t) = 1$ there exists a leaf v in layer L_t in B such that $\mathrm{reach}_{u_{j_{\ell-1}}^{\ell-1},v}(\sigma_{h(\ell-1)+1},\ldots,\sigma_t) = 1$ and, therefore, in G there exists a path P^ℓ from $u_{j_{\ell-1}}^{\ell-1}$ to a leaf v such $\mathrm{sat}_{P^\ell}(\sigma) = 1$. By concatenating the paths $P^1,\ldots,P^\ell$, we obtain a path P in G from u_0 to a leaf for which $\mathrm{sat}_P(\sigma) = 1$; thus, B accepts σ.

To bound $|L'_i|$, recall that a vertex in layer i of T is either $u_{0,j_1,\ldots,j_i}$ or leafs of the form $v_{0,j_1,\ldots,j_{i-1},j}$, where $h(i-1)+1 \leq j \leq h(i)$. Thus $|L'_i| \leq \left(\prod_{1\leq j\leq i-1} w(h(i))\right)\left(w(h(i)) + h(i) - h(i-1)\right)$. □

Next, we construct a strong evolving CDS protocol for LINBPs.

Theorem 4.5. *Let $B = (G = (V,E), u_0, \mu)$ be an LINBP implementing an evolving predicate $f : \{0,1\}^* \to \{0,1\}$ and $h : \mathbb{N} \to \mathbb{N}$ be an increasing function, where $h(0) = 0$. Then, there exists a strong evolving CDS protocol realizing f in which the message of server q_t in generation i, i.e., $h(i-1)+1 \leq t \leq h(i)$, consists of the messages of q_t in the $\left(\prod_{1\leq j\leq i-1} w(h(j))\right)(w(h(i)) + h(i) - h(i-1))$ strong CDS protocols realizing the predicates of the edges between layer $i-1$ and layer i in the GIDT constructed in Construction 4.3.*

Proof. We use Construction 4.3 to transform B into an equivalent GINDT T with the specified h (The choice of h depends on each LINBP itself. It is a trade-off between a larger number of levels in GINDT and a larger number of servers in each generation.). Next, we apply Construction 4.1 to T to obtain a strong evolving CDS protocol realizing T. Finally, by Claim 4.4 and Claim 4.2, we derive a strong evolving CDS protocol realizing B with the messages as stated in the theorem. □

4.1 Application of Evolving Strong CDS Protocols for LINBP with Bounded Width

Next we demonstrate that an LINBP-based construction yields non trivial upper bounds on (strong) evolving CDS for evolving predicates represented by LINBP's of width $O(2^{0.2499t})$. To compare, the similar LIBP-based construction for evolving secret sharing yields non-trivial bounds only for LIBP's of width $O(2^{0.15t})$.

Theorem 4.6. *Let $0 < \epsilon < 0.25$ denote a constant, and B a LINBP of width $w(t) \leq 2^{\epsilon t}$ implementing an evolving predicate $f : \{0,1\}^* \to \{0,1\}$. Then f has a strong evolving CDS with message complexity $O(2^{dt})$ for a constant $d < 1$.*

Proof. We apply Theorem 4.5 to B and $h(i) = c^i$. Fix a server q_t and let i be the generation of q_t, that is, $c^{i-1} + 1 \leq t \leq c^i$, in particular

$$c^i \leq ct. \tag{1}$$

By Theorem 4.5, the number of strong CDS messages for the relevant predicates (and different

CDS instances) that p_t holds is

$$\left(\prod_{1\le j\le i-1} w(h(j))\right)(w(h(i)) + h(i) - h(i-1)) \le \tag{2}$$

$$\left(\prod_{1\le j\le i-1} w(h(j))\right)(w(h(i)) + ct) \tag{3}$$

$$= O\left(\prod_{1\le j\le i} 2^{\epsilon c^j}\right) = O\left(2^{\epsilon \sum_{1\le j\le i} c^j}\right)$$

$$\le O\left(2^{\epsilon c^{i+1}/(c-1)}\right) \le O\left(2^{\epsilon c^2 t/(c-1)}\right), \tag{4}$$

where the first and the last inequality is from (1). To realize the predicates of the edges, we use the best known k-party CDS protocols Theorem 2.1 (applied with $k \le h(i) - h(i-1) \le ct$, modified to be strong via Construction 3.1, which does not impose an overhead which impacts our bound. Thus, the size of the message of q_t is

$$2^{\epsilon c^2 t/(c-1)} \cdot 2^{\tilde{O}(\sqrt{tc})} = 2^{\epsilon c^2 t/(c-1)+o(t)} \tag{5}$$

We choose the optimum $c = 2$ and get by Equation (5) that for every LINBP of width $w(t) = 2^{\epsilon t}$ for $\epsilon < 0.25$ the message size of p_t is less than $O(2^{dt})$ for a constant $d < 1$. □

5 Improved Evolving Secret-Sharing Schemes for a Class of Evolving Access Structures via Strong Evolving CDS Protocols

In this section, we present an optimized evolving secret-sharing schemes construction for a specific class of evolving access structures, so called partite access structures, based on (strong) evolving CDS protocols. The scheme's efficiency is beyond what [1]'s IBP (infinite branching program)-based protocol allows for.

The class of functions in question is obtained from evolving predicates, similarly to partite access structures derived from (not necessarily monotone) functions in a way that a CDS protocol immediately translates into a secret sharing scheme for the function essentially with no overhead [2].

Definition 5.1. [**Evolving Partite Access Structure**] Let $f : \{0,1\}^* \to \{0,1\}$ denote an evolving predicate as in Definition 2.2 for secret domain S. We define an evolving AS $\mathcal{A}_f$ corresponding to it as follows. $A_f : \{0,1\}^* \to \{0,1\}$ is defined over the set of parties $p_{1,0}, p_{1,1}, p_{2,0}, p_{2,1}, \ldots$ with the minterm set $M = \{p_{i,0}, p_{i,1} | i \in \mathbb{N}^+\} \cup \{p_{1,x_1}, \ldots, p_{t,x_t} | t \ge 1, f(x_1, \ldots, x_t) = 1, \forall i < t, f(x_1, \ldots, x_i) = 0\}$. We refer to such AS's as partite evolving AS's, and to $\mathcal{A}_f$ as the partite evolving AS corresponding to f.

On a high level, our secret sharing for $\mathcal{A}_f$ is a black box reduction to an evolving strong CDS for f. We note that it beats [1], when the edge predicates in their main constructions are instantiated by generic schemes. That is, either the best known worst case [3] or the best known directed st-conn

based secret sharing for a graph derived from the LIBP. [9] If we instantiate those predicates with the best known CDS-based secret sharing schemes (as the edge predicates are in fact finite partite functions), the result has similar complexity to what we get in the following construction, when instantiated with the strong CDS in Section 4.

Theorem 5.1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ denote an evolving predicate, and let $\mathcal{A}_f$ denote the partite evolving AS, and $\mathcal{P}$ denote a strong evolving CDS protocol for f for secret domain S and message complexity $cds(t)$. Then $\mathcal{A}_f$ has a secret sharing scheme with share complexity $sc(t) = cds(\lceil t/2 \rceil) + |log(S)|$.*

We prove the theorem by a simple direct construction.

Construction 5.2 ($\Pi_{f,CDS}$ - an evolving secret-sharing scheme for $\mathcal{A}_f$)**.**

- **Input:** $s \in S$.
- Let $\mathcal{P}_{CDS} = (\mathrm{ENC}_{CDS}, \mathrm{DEC}_{CDS})$ denote a strong evolving CDS protocol for f, with secret domain S, and randomness domain sequence $\{R_t\}_{t\in\mathbb{N}}$.
- **Share**$((t,b), s, r_1, \ldots, r_{t-1})$:
  - When party $p_{t,b}$ arrives :
  - If $b = 0$:
    - Pick a random $r_t \in Z_{|S|}$. Let $\mathbf{Sh}^1_{(t,0)} = r_t + s \mod |S|$.
    - Sample CDS randomness $r'_t \leftarrow R_t$.
  - If $b = 1$: let $\mathbf{Sh}^1_{(t,1)} = r_t$.
  - Let $\mathbf{Sh}^2_{t,b} = Enc_t(b, s, r'_1, \ldots, r'_t)$.
  - Give $p_{t,b}$ the share $\mathbf{Sh}_{t,b} = (\mathbf{Sh}^1_{t,b}, \mathbf{Sh}^2_{t,b})$.

Claim 5.3. $\Pi_{f,CDS}$ is an evolving secret-sharing scheme for $\mathcal{A}_f$ as claimed to exist in Theorem 5.1

Proof. First we prove the correctness of the scheme. Let A denote a (finite) qualified set of parties. If $p_{t,0}, p_{t,1} \in A$ for some $t \geq 1$, the secret can be reconstructed via $s = \mathbf{Sh}^1_{t,0} + \mathbf{Sh}^1_{t,1} \mod |S|$. Otherwise, there exist $x \in \{0,1\}^t$ with $f(x) = 1$ such that $p_{1,x_1}, \ldots, p_{t,x_t} \in A$. Thus, the parties can reconstruct the secret from $\mathrm{DEC}_t(\mathbf{Sh}^2_{1,x_1}, \ldots, \mathbf{Sh}^2_{t,x_t})$, by correctness of $\mathcal{P}_{CDS}$.

Next we prove the security of the scheme. Let B denote a (finite) unqualified set of parties. It does not contain $p_{t,0}, p_{t,1}$ for any $t \geq 1$, or it would be qualified. Thus, the $\mathbf{Sh}^1$ parts are all random and independent (among themselves and of the $\mathbf{Sh}^2$'s) in $\mathbb{Z}_{|S|}$ (for all s). By definition of $f, \mathcal{A}_f$, there exists no $\mathbf{x} = (x_1, \ldots, x_t)$ such that $f(\mathbf{x}) = 1$, and $\{p_{i,x_i} | i \in [t]\} \subseteq A$. Thus, the $\mathbf{Sh}^2$'s do not reveal any information due to the fact that $\mathcal{P}_{CDS}$ is strong. □

[9] It beats the upper bound resulting from these protocols for a wide range of BP width parameters, and often beats the actual resulting complexity resulting from either instantiation.

6 Lower bounds for (strong) evolving CDS

In this section, we demonstrate that similarly to evolving secret-sharing schemes, (strong) evolving CDS can not generally be achieved with non-trivial message complexity of $2^{ct+o(t)}$ for constant $c < 1$. We prove the result in two steps. First, we craft a partite evolving secret-sharing scheme A_f for which we can prove a tight lower bound on share complexity. Then, we use Theorem 5.1 to obtain a lower bound for strong evolving CDS. See Section 1.3.2 for more discussion on the high level proof structure, and comparison with prior work.

Theorem 6.1. *There exists an evolving predicate $f : \{0,1\}^* \to \{0,1\}$, such that any strong evolving CDS for it, and secret domain $S = \{0,1\}$ does not have a message complexity bound of the form $st(t) = O(2^{ct})$.*

Proof of Theorem 6.1. We construct f in an iterative manner, along with a lower bound $l(t) \in 2^{t-o(t)}$ on the share complexity of $\mathcal{A}_f$. We specify $\mathcal{A}_f$ by the sequence of minterms, which determines the entire (partite) evolving AS. Note that the minterms which are not of the form $\{p_{i,0}, p_{i,1}\}$ are specified by some $\mathbf{b} \in \{0,1\}^t$, as $A_{\mathbf{b}} = \{p_{i,\mathbf{b}_i} | i \in [t]\}$, because the AS is partite.
[10]. Let us (arbitrarily) set $t_1 = 16$.

- For every $t \geq 1$, add $\{p_{t,0}, p_{t,1}\}$ to the set of minterms of $\mathcal{A}_f$.
- Let $t_1 = 16$,$pref_1 = 1^{\log(t_1)} = 1^4$.
- For $i \geq 1$:
  - Add the following minterms: for every $v \in \{0,1\}^{t_i - |pref_i|}$, add
  $$A_v = \{p_{i,(pref_i||v)_1}, \ldots, p_{t_i,(pref_i||v)_{t_i}}\} \cup \{p_{(t_i+1,0)}, \ldots, p_{(t_i+1+v,0)}\}$$
  .[11]
  - Set $pref_{i+1} = 1^{t_i}$ and fix $t_{i+1} = 2^{t_i}$.[12]

First, we observe that $\mathcal{A}_f$ specified above is indeed a partite evolving AS, and all sets added are indeed minterms (rather than being implied by previously added minterms). This easily follows by induction. Clearly, for every i, the A_v's added at step i are all incomparable (as sets). Now, for $i > 1$, we add minterms A_v containing the set $\{p_{j,1} | j \in [t_i]\}$ as a subset, while all previously added minterms either of the form $\{p_{l,0}, p_{l,1}\}$ or ones that contained $p_{t_j+1,0}$ for some $t_j < t_i - 1$ (as part of the 'B-set'), and did not contain $p_{t_j+1,1}$.

Claim 6.2. For $\mathcal{A}_f$ constructed above, for every evolving secret sharing scheme Sh, and for infinitely many values $t \geq 1$, p_t size in Sh of at least $\ell(t)$, for a function $\ell(t) = 2^{t/2-o(t)}$.

[10] f is as uniquely induced by $\mathcal{A}_f$ we define.
[11] In $t_i + 1 + v$ we interpret v as a binary number (possibly with leading zeros).
[12] Other large functions would also do. Picking larger growth rate would result in a larger bound. In fact, we could arbitrarily strengthen the bound proving for any $g(t) = \omega(1)$, strong evolving CDS share complexity for f can not be $2^{t-g(t)}$ for all $t > t_0$ for some t_0.

Proof of Claim. Fix some $i \geq 1$. Then the claim follows from Theorem 1.6, applied to the set $A = A^i = \{p_{j,b} | j \in [t_i], b \in \{0,1\}\} \setminus \{p_{i,0} | i \in [\log(t_i)]\} \subseteq [t'_i]$, induced by the independent sequence comprised of the A_v's added at time i (which are subsets of A^i) and $B = B^i = \{p_{t_i+1,0}, \ldots, p_{t_i+1+\ldots,0}\}$ (up to reordering of parties). Indeed, it is an independent sequence, as all $\ell = 2^{t'_i/2-log(t'_i)+1}$ sets in the sequence are minterms of $\mathcal{A}_f$ (see above). Thus, the total share complexity for $\mathcal{A}_f \cap 2^{[t'_i]}$ is lower bounded by $\ell - 1$ (for every valid evolving secret-sharing schemes for $\mathcal{A}_f$). Therefore, at least one of the members of A, gets a share of size

$$\geq 2^{t'_i/2-log(t'_i)+1}/|A| = 2^{t'_i/2-log(t'_i)+1}/(t'_i - log(t'_i) + 1) = 2^{t'_i/2-o(t'_i)}.$$

Let us denote the highest index in A^i of such a party by max_i. Now, even if $max_i = t'_i$, its dependence of share size on its index is of the form $sc(t) = 2^{t/2-o(t)}$. Finally, there must exist an infinite sequence of parties with this share complexity bound. As a simple way to see this, we observe that if all max_i's belonged to a finite set B of party indices. For a sufficiently large j, for every $p_i \in B$, we have that $|sh_i| < 2^{|A^j|/2-log(|A^j|+2)}/|A^j|$. We conclude that $max_j \notin B$, contradicting the fact that B is the set of all max_i values. □

We are now ready to complete the proof of Theorem 6.1. Here we show that for every evolving secret sharing scheme **Sh** for $\mathcal{A}_f$ above, infinitely many parties t_i, have share size at least $2^{t_i/2-o(t_i)}$. By Theorem 5.1, a strong evolving CDS protocol for f with message complexity $c(t)$, then the share complexity of $\mathcal{A}_f$ is at most $c(t/2)+O(1)$ for all t. Assuming for contradiction $c(t) \leq 2^{ct+o(t)}$ for some $c < 1$, we obtain an evolving secret-sharing scheme with share complexity $c(t'_i/2)+O(1) = 2^{ct'_i/2-o(t'_i)}$ which is strictly below the lower bound above for all sufficiently large t_i's as above, leading to a contradiction. □

It is an interesting open question to obtain a similar result for (non-strong) evolving CDS. We note that the lower bound for strong evolving CDS can already be viewed as a certain clue that standard evolving CDS may be hard (in terms of requiring message complexity $2^{t-o(t)}$ in the worst case, as the best known CDS we know how to construct 4 implicitly puts forward a CDS construction that can be made strong with very small complexity overhead. Although we could not currently find a generic "black box" transformation, it could be the case that concrete evolving CDS constructions could be easy to come up with.

Bibliography

[1] Bar Alon, Amos Beimel, Tamar Ben David, Eran Omri, and Anat Paskin-Cherniavsky. New upper bounds for evolving secret sharing via infinite branching programs. In *Theory of Cryptography: 22nd International Conference, TCC 2024*, page 548–580, Berlin, Heidelberg, 2024. Springer-Verlag.

[2] Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1553–1566. ACM, 2023.

[3] Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1.5^n. In *CRYPTO 2021*, volume 12827, pages 627–655. Springer, 2021.

[4] Amos Beimel. Lower bounds for secret-sharing schemes for k-hypergraphs. In Kai-Min Chung, editor, *4th Conference on Information-Theoretic Cryptography, ITC 2023, June 6-8, 2023, Aarhus University, Aarhus, Denmark*, volume 267 of *LIPIcs*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[5] Amos Beimel and Hussien Othman. Evolving ramp secret-sharing schemes. In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Proceedings*, pages 313–332, Germany, January 2018. Springer Verlag.

[6] Amos Beimel and Hussien Othman. Evolving ramp secret sharing with a small gap. In *EUROCRYPT 2020*, volume 12105 of *LNCS*, pages 529–555, 2020.

[7] Amos Beimel, Hussien Othman, and Naty Peter. Quadratic secret sharing and conditional disclosure of secrets. In *CRYPTO 2021*, pages 748–778, 2021.

[8] Amos Beimel and Naty Peter. Optimal linear multiparty conditional disclosure of secrets protocols. In *ASIACRYPT 2018*, volume 11274 of *LNCS*, pages 332–362, 2018.

[9] Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. *SIAM J. Discret. Math.*, 22(1):360–397, 2008.

[10] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO '88*, volume 403 of *LNCS*, pages 27–35, 1988.

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

[12] Carlo Blundo, Alfredo De Santis, Roberto de Simone, and Ugo Vaccaro. Tight bounds on the information rate of secret sharing schemes. *Des. Codes Cryptogr.*, 11(2):107–122, 1997.

[13] Shion Samadder Chaudhury, Sabyasachi Dutta, and Kouichi Sakurai. Ac^0 constructions of secret sharing schemes - accommodating new parties. In *NSS 2020*, volume 12570 of *LNCS*, pages 292–308, 2020.

[14] Qi Cheng, Hongru Cao, Sian-Jheng Lin, and Nenghai Yu. A construction of evolving k-threshold secret sharing scheme over a polynomial ring. *arXiv preprint arXiv:2402.01144*, 2024.

[15] László Csirmaz. The size of a share must be large. volume 10, page 223–231, Berlin, Heidelberg, sep 1997. Springer-Verlag.

[16] Paolo D'Arco, Roberto De Prisco, and Alfredo De Santis. Secret sharing schemes for infinite sets of participants: A new design technique. *Theor. Comput. Sci.*, 859:149–161, 2021.

[17] Paolo D'Arco, Roberto De Prisco, Alfredo De Santis, Angel Pérez del Pozo, and Ugo Vaccaro. Probabilistic Secret Sharing. In *MFCS 2018*, volume 117 of *LIPIcs*, pages 64:1–64:16, 2018.

[18] Yvo Desmedt, Sabyasachi Dutta, and Kirill Morozov. Evolving perfect hash families: A combinatorial viewpoint of evolving secret sharing. In *Cryptology and Network Security*, pages 291–307, 2019.

[19] Sabyasachi Dutta, Partha Sarathi Roy, Kazuhide Fukushima, Shinsaku Kiyomoto, and Kouichi Sakurai. Secret sharing on evolving multi-level access structure. In Ilsun You, editor, *Information Security Applications*, pages 180–191, Cham, 2020. Springer International Publishing.

[20] Danilo Francati and Daniele Venturi. Evolving secret sharing made short. Cryptology ePrint Archive, Paper 2023/1534, 2023. https://eprint.iacr.org/2023/1534.

[21] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *Journal of Computer and System Sciences*, 60(3):592–629, 2000.

[22] Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

[23] Mauricio Karchmer and Avi Wigderson. On span programs. *[1993] Proceedings of the Eigth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993.

[24] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. *IEEE Trans. Inf. Theory*, 64(6):4179–4190, 2018.

[25] Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: Dynamic thresholds and robustness. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 379–393, Cham, 2017. Springer International Publishing.

[26] Tianren Liu, Vinod Vaikuntanathan, and Wee Hoeteck. Towards breaking the exponential barrier for general secret sharing. In *Advances in Cryptology – EUROCRYPT 2018*. Springer International Publishing, 2018.

[27] Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 567–596, 2018.

[28] Noam Mazor. A lower bound on the share size in evolving secret sharing. In Kai-Min Chung, editor, *4th Conference on Information-Theoretic Cryptography, ITC 2023, June 6-8, 2023, Aarhus University, Aarhus, Denmark*, volume 267 of *LIPIcs*, pages 2:1–2:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[29] Ryo Okamura and Hiroki Koga. New constructions of an evolving 2-threshold scheme based on binary or d-ary prefix codes. In *2020 International Symposium on Information Theory and Its Applications (ISITA)*, pages 432–436, 2020.

[30] Naty Peter. Evolving conditional disclosure of secrets. In Elias Athanasopoulos and Bart Mennink, editors, *Information Security*, pages 327–347, Cham, 2023. Springer Nature Switzerland.

[31] Kittiphop Phalakarn, Vorapong Suppakitpaisarn, Nuttapong Attrapadung, and Kanta Matsuura. Evolving homomorphic secret sharing for hierarchical access structures. In *Advances in Information and Computer Security: 16th International Workshop on Security, IWSEC 2021, Virtual Event, September 8–10, 2021, Proceedings*, page 77–96, Berlin, Heidelberg, 2021. Springer-Verlag.

[32] Adi Shamir. How to share a secret. In *Communications of the ACM, 22*, pages 612–613, 1979.

[33] Chaoping Xing and Chen Yuan. Evolving secret sharing schemes based on polynomial evaluations and algebraic geometry codes. *IEEE Transactions on Information Theory*, 70(5):3718–3728, 2024.

[34] Wei Yan, Sian-Jheng Lin, and Yunghsiang S. Han. A new metric and the construction for evolving 2-threshold secret sharing schemes based on prefix coding of integers. *IEEE Transactions on Communications*, 71(5):2906–2915, 2023.