Secret-Sharing Schemes for High Slices

Amos Beimel[1], Oriol Farràs[2], Or Lasri[1], and Oded Nir[3]

[1] Ben-Gurion University of the Negev, Be'er-Sheva, Israel
[2] Universitat Rovira i Virgili, Tarragona, Spain
[3] Tel Aviv University, Tel Aviv, Israel
amos.beimel@gmail.com, oriol.farras@urv.cat, orshlomo@post.bgu.ac.il, odednir123@gmail.com

Abstract. In a secret-sharing scheme, a secret is shared among n parties such that the secret can be recovered by authorized coalitions, while it should be kept hidden from unauthorized coalitions. In this work we study secret-sharing for k-slice access structures, in which coalitions of size k are either authorized or not, larger coalitions are authorized and smaller are unauthorized. Known schemes for these access structures had smaller shares for small k's than for large ones; hence our focus is on "high" $(n-k)$-slices where k is small.

Our work is inspired by several motivations: 1) Obtaining efficient schemes (with perfect or computational security) for natural families of access structures; 2) Making progress in the search for better schemes for general access structures, which are often based on schemes for slice access structures; 3) Proving or disproving the conjecture by Csirmaz (J. Math. Cryptol., 2020) that an access structures and its dual can be realized by secret-sharing schemes with the same share size.

The main results of this work are:

Perfect schemes for high slices. We present a scheme for $(n-k)$-slices with information-theoretic security and share size $kn\cdot 2^{\tilde{O}(\sqrt{k \log n})}$. Using a different scheme with slightly larger shares, we prove that the ratio between the optimal share size of k-slices and that of their dual $(n-k)$-slices is bounded by n.

Computational schemes for high slices. We present a scheme for $(n-k)$-slices with computational security and share size $O(k^2 \lambda \log n)$ based on the existence of one-way functions. Our scheme makes use of a non-standard view point on Shamir secret-sharing schemes that allows to share many secrets with different thresholds with low cost.

Multislice access structures. $(a:b)$-*multislices* are access structures that behave similarly to slices, but are unconstrained on coalitions in a wider range of cardinalities between a and b. We use our new schemes for high slices to realize multislices with the same share sizes that their duals have today. This solves an open question raised by Applebaum and Nir (Crypto, 2021), and allows to realize hypergraph access structures that are chosen uniformly at random under a natural set of distributions with share size $2^{0.491n+o(n)}$ compared to the previous result of $2^{0.5n+o(n)}$.

This document is the full version of [BFLN24], published at TCC 2024.

1 Introduction

Secret-sharing schemes, introduced by Shamir [Sha79] and Blakley [Bla79], is a pivotal cryptographic primitive that has many applications in cryptography and in neighboring fields (see, e.g., survey works of [Bei11,ADH17,CSNN24] for details about such applications). In a secret-sharing scheme, a dealer that holds a secret shares it among n parties, by sending each party a single message (called a share). It is required that predefined authorized coalitions will be able to recover the secret from their shares and that the secret will remain hidden from all unauthorized coalitions. A scheme is called *perfect* when the secret is kept information-theoretically hidden from unauthorized sets (i.e. they cannot learn anything about the secret from their shares even if they are computationally unbounded); it is called *computational* if secrecy is held against parties that are computationally bounded. The collection of authorized coalitions is called an *access structure*, and it can be captured by a monotone function $f : \{0,1\}^n \to \{0,1\}$ that outputs 1 on an input $x \in \{0,1\}^n$ iff x is the characteristic vector of an authorized set.

The most important efficiency measure for secret-sharing schemes is the size of the shares dealt to the participating parties. Hence, the goal of many research works has been to realize all (general) n-party access structures with small shares. Towards this end, modern schemes (following the seminal work of Liu and Vaikuntanathan [LV18]) typically first realize restricted families of access structures with non-trivially small shares and then compose them, in some "sophisticated" way, to get better schemes for general access structures. Improving the share size of such restricted families of access structure became a relatively central problem in the field. Among the families used in the above-mentioned paradigm, we can list k-slices (also knwon as k-uniform access structures) and $(a : b)$-*multislices.* A k-slice function can output arbitrary values for inputs of Hamming weight k, and must output 0 on lighter inputs and 1 on heavier ones. $(a : b)$-*multislices* are monotone functions that are unconstrained on inputs of Hamming weight between a and b, but must take the value 0 on lighter inputs, and the value 1 on heavier inputs. Note that a $(k : k)$-multislice is a k-*slice.*

Despite the growing importance of these access-structure families, works that have studied them so far have been, in some sense, incomprehensive, as they mainly focused on the regime where $k \ll n$. For example, there are at least a dozen papers dealing with secret-sharing schemes for 2-slices (also known as *forbidden graph access structures* (see, e.g., [SS97,BIKK14,BFMP22]). To the best of our knowledge, no previous papers study $(n-2)$-slices. Moreover, the best perfect schemes for slices have much smaller shares when k is small ("low slices"), compared to when k is large ("high slices"). For computational schemes the situation is even worse, as we do not know of any work that studied computational secret-sharing schemes for slices based on most basic assumption of the existence of one-way functions (OWF).[4] We therefore bring forward the following questions:

- Can high slices be realized by a perfect scheme with the same share sizes as low slices?
- Can better schemes for high slices help improve schemes for general access structures?
- Can natural families of functions like slices be realized with smaller shares assuming OWFs exist?

Before we move on to survey the literature regarding the topics discussed so far, we note that these questions are also closely related to the concept of secret-sharing *duality.* In the notation of functions, the dual of an access structure $f : \{0,1\}^n \to \{0,1\}$ is the function f^* that satisfies $f^*(x) = 1 - f(\overline{x})$, where $\overline{x}$ is the string for which $\overline{x}_i = 1 - x_i$ in every $i \in [n]$. We observe that if f is monotone then so is f^*, that for every function f it holds that $(f^*)^* = f$, that the duals of k-slices are $(n-k)$-slices, and that the duals of $(a : b)$-multislices are $(n-b : n-a)$-multislices. For linear and multi-linear secret-sharing schemes (see Definition 2.2), the optimal share size of every function and its dual are identical [Gál95,Feh98,FHKP17], but it is not known whether this property holds for general schemes. In his work from 2020, Csirmaz [Csi20] focused on this question and formalized the following conjecture:

Conjecture 1.1 (Csirmaz's conjecture). The optimal share size per bit of the secret (also known as the *information ratio*) of primal and dual access structures is equal.

Csirmaz showed that in a relaxed model of secret sharing (where errors in recovery or security may occur with negligible probability), the conjecture is false. It is entirely unclear whether this says anything about duality in the standard (error-free) model when sharing one-bit secrets. In fact, we can neither exclude the possibility that the conjecture is true nor the possibility of an exponential gap between the share size of access structures and their duals. On the positive side, it is known that for simple functions such as thresholds the dual and primal share sizes are equal for large enough secrets, and Bogdanov [Bog23] recently proved that the optimal share size of 2-thresholds and $(n-1)$-thresholds is exactly the same for every n. Hence, studying duality of more complex families of functions seems like a natural next step in better understanding Csirmaz's conjecture. We ask the following question about the share size of slices:

- Can we bound the gap between the share sizes of slices and their duals? I.e., given a scheme for k-slices with share size L, can we realize their dual $(n-k)$-slices with shares of size $L \cdot t(n,k)$ where $t(n,k)$ is small?

[4] Computational schemes for low slices follow from combining results of [ABF$^+$19,ABI$^+$23b].

1.1 Related Work

Perfect schemes for general access structures. The first perfect schemes for general access structures that were introduced by Ito, Saito and Nishizeki. [ISN87] had shares of size $O(2^n)$. A generalization of these schemes was presented by Benaloh and Leiscter [BL88], showing that if a function can be computed by a monotone formula of size L then it can be realized by a secret-sharing scheme with total share size L. In [KW93], it was shown that the share size of any function can also be tied to its monotone span-program complexity, but this is still $O(2^n)$ in the worst case. A breakthrough result of Liu and Vaikuntanathan [LV18] followed decades later, describing a scheme with share size $O(2^{0.994n})$. Since then, the state of the art has further improved several times [ABF+19,ABNP20,AN21]. In the latest among these developments, Applebaum and Nir [AN21] showed how to realize general access structures with shares of size $1.5^{n+o(n)}$. On the lower bound front, Csirmaz [Csi97] described an access structure that requires shares of size $\Omega(n/\log n)$, and it was proved in [ABN+22] that the modern techniques following the breakthrough of [LV18] cannot realize all access structures with shares smaller than $2^{o(n/\log^2 n)}$.

Perfect Schemes for Slices and Multislices. Secret-sharing schemes for slice access structures, also called *uniform access structures*, were previously studied in several works as [AA18,BKN18,BP18,LV18,ABF+19,AN21,ABN+22]. There exists a simple scheme that realizes every k-slice f by taking a monotone DNF or CNF formulas for f and applying the formula-to-scheme transformation of [BL88]. For one-bit secrets,[5] this scheme has shares of size $\binom{n-1}{k-1}$ for k-slices and $\binom{n-1}{n-k}$ for $(n-k)$-slices for $k \leq n/2$. The best-known upper bounds on the share size of slices in literature outperform the naive scheme above in some regimes, as detailed in the following Fig. 1. Prior to this work (see the bounds stated in Fig. 1), there exists a gap between the share sizes of slices and their duals. When k is constant, k-slices have share sizes of $n^{o(1)}$ while their dual $(n-k)$-slices have share sizes of $O\left(\binom{n-1}{n-k}\right) = O(n^{k-1})$. When $k = \log n$ (a regime is relevant for realizing multislices and general functions) the gap will be between $n^{O(\log\log n)}$ for low slices and $O(n^{\log n})$ for high ones. We also note that the multi-linear scheme of [AA18] for k-slices has information ratio $2^{O(k)}$ for secrets with size that is double-exponential in n. By the duality closure properties for multi-linear schemes, this implies that there exists a scheme for $(n-k)$-slices with the same information ratio.

For multislice access structures, the situation is similar. Applebaum and Nir [AN21] designed a scheme for $(a : b)$-multislices as a stepping stone for

[5] For long secrets it is sometimes known how to realize schemes with smaller share sizes per secret bit (better *information ratio*) with amortization techniques. The share size of mentioned scheme based on formulas can be improved by a factor of $\log n$ for moderately long secrets and $k \leq n/2$ [EP97,Bei23], and a k-slice scheme of [ABF+19] has information ratio k^2 for secrets of size that is doubly-exponential in n.

The share size of perfect schemes for k-slices and $(n-k)$-slices

Slice height	Below $\log n$	Between $\log n$ and $n/\log n$	Between $n/\log n$ and $n-n/\log n$	$(n-k)$-slices for $k \leq n/\log n$
Upper bounds	$2^{O(k)+\tilde{O}(\sqrt{k \log n})}$ [AA18]	$kn \cdot 2^{\tilde{O}(\sqrt{k \log n})}$ [ABF$^+$19]	$2^{\tilde{O}(\sqrt{n})}$ [LV18]	$kn \cdot 2^{\tilde{O}(\sqrt{k \log n})}$ Theorem 3.5, compared to $O(n^{k-1})$ in the formula-based scheme
Lower bounds	$\Omega(\log n)$ [KN90,CCX13,BGK16]			

Fig. 1. The best-known bounds on the share size of perfect secret-sharing schemes for k-slices for $1 < k < n-2$. For $k = 1$ there exist simple schemes with share size $\log n$, and for $k = n-2$ shares of size $O(\sqrt{n})$ can be obtained by taking the dual scheme of the 2-slice scheme of [GKW15]. The $\Omega(\log n)$ lower bound by [KN90,CCX13] was proved for the 2-threshold function (which is also a 2-slice function). The same bound was later proved in [BGK16] for all k-slices. The borders between the ranges of parameters are written without asymptotical notation for better readability (e.g., should be $\Theta(\sqrt{n})$ instead of $\sqrt{n}$).

schemes for general functions with shares of size $\min\{\binom{b}{\geq a} \cdot 2^{o(n)}, 2^{0.585n+o(n)}\}$, where $\binom{b}{\geq a} := \sum_{a \leq i \leq b} \binom{b}{i}$. It is not hard to see that this scheme is not "balanced" with respect to duality. For example, the share size for $(0 : 0.1n)$-multislices is $2^{0.1n+o(n)}$ while that of their dual $(0.9n : n)$-multislices is $2^{\mathrm{H}_2(0.9)n+o(n)} > 2^{0.45n}$, where H_2 is the binary entropy function.

Computational Secret-Sharing Schemes Based on OWF. Computational secret-sharing schemes (CSSS) can be based on a variety of cryptographic hardness assumptions. In this work, we will focus on the most basic one: the existence of one-way functions (OWFs). In the computational setting, the efficiency of schemes will also be measured with respect to a security parameter λ.[6] Yao [Yao89] (see also [VNS$^+$03]) was the first to consider secret-sharing schemes in the computational setting. He showed that assuming the existence of one-way functions, any function that can be computed by a monotone circuit with C wires can be realized by a CSSS with share size $O(\lambda C)$. [7]

[6] In the computational setting, the share size may also be reduced by using public information.

[7] Alternatively, with a CSSS with shares of size $O(\lambda)$ and public information of size $O(\lambda C)$. As mentioned before, in the information-theoretic setting, a similar result is only known for monotone formulas [BL88].

Krawczyk [Kra94] showed how to share large secrets of size S according to a k-threshold function with shares of size $|S|/k+\lambda$, thus bypassing an information-theoretic lower bound [KGH83] that states that shares cannot be smaller than the secret size. In this example, as opposed to the perfect schemes mentioned so far, the share sizes decrease when the cardinality k of the authorized sets increases.

In the latest exciting study of computational schemes by Applebaum et al. [ABI$^+$23b], they introduced new efficient schemes based on one-way functions for several families of access structures, including DNF formulas with long terms and CDS protocols (which are essentially a special class of slice functions, see discussion below). Their k-server CDS protocols have messages of size $\lambda + O(1)$ and poly$(t(\lambda))$-security (for a binary domain of inputs). By the connections between CDS protocols and secret-sharing schemes for slices [ABF$^+$19,AA18], it can be shown (similarly to the proof in Section 4) that this implies that k-slices can be realized with shares of size $O\left(\lambda \log n \cdot \min\left\{kn, 2^{O(k)}\right\}\right)$ for $k \leq \sqrt{n}$ or $k \geq n - \sqrt{n}$ if OWF exist. Unlike the scheme of Krawczyk for thresholds, here the share size grows with k, the cardinality of the minimal authorized sets, and high slices are more expensive than low ones. Constructing computational secret-sharing schemes based on one-way functions for additional families of access structures, or even for all access structures, is an interesting open problem.

Besides the results discussed so far that are based on one-way functions, some schemes in the literature were based on stronger assumptions. In [ABI$^+$23b], they designed several such schemes. Under the RSA assumption, they describe a CSSS that, given an arbitrary access structure f, represented by a truth table of size $N = 2^n$, produces shares of size poly(n) in time $\tilde{O}(N)$. Weaker results are obtained under the decisional Diffie-Hellman and the decisional bilinear Diffie-Hellman assumptions. Under the RSA assumption, they also realize monotone CNF formulas with share size polylog(m), where m is the number of clauses in the CNF formula. When considering $(n-k)$-slices that can be computed by a CNF with $O(n^k)$ clauses, the RSA based scheme with shares of size poly$(k \log n)$. In [KNY17], they give a construction of a computational secret-sharing scheme for any monotone function in NP assuming witness encryption for NP and the existence of one-way functions.

1.2 Our Results

We present several secret-sharing schemes for high slice functions, aiming to narrow or close as many gaps as possible between the share size of low slices and that of high slices. Our computational scheme for high slices will perform even better than its counterpart for low slices.

We prove the following theorem for perfect schemes:

Theorem 1.2 (Perfect Schemes for High Slices). *Let $k \leq n/2$ be positive integers. For every $(n-k)$-slice function f, there exists a secret-sharing scheme realizing f with share size $kn \cdot 2^{\tilde{O}(\sqrt{k \log n})}$.*

Our scheme closes the current gap in share sizes between slices and their duals when k is logarithmic in n (share size of $n^{O(\log\log n)}$ in both cases), and narrows it down substantially when k is constant ($n^{1+o(1)}$ compared to $n^{o(1)}$ for low constant-k slices). We note that given any constant integer k, our scheme for $(n-k)$-slices even outperform the scheme by Applebaum et al. [ABF$^+$19] that only works for long secrets of size at least $2^{n^{n-k}}$, and has shares of size $O(n^2)$ per secret bit.

We also present a scheme for $(n-k)$-slices with a simpler structure that proves the following theorem:

Theorem 1.3 (Duality and Slices). *For every two integers $k < n$, if there exists an n-party secret-sharing scheme for k-slices with share size L, then there exists an n-party secret-sharing scheme for $(n-k)$-slices with share size $L \cdot n$.*

Our scheme works for every k, and so it allows to realize high slice functions with low ones or the other way around. Thus, by Theorem 1.3 the ratio between the share size of slice functions and their duals is bounded by n in both directions. We remark that for a given $(n-k)$-slice, our construction uses a k-slice that is *not* its dual.

Next, we present a computational scheme for high slices, which implies the following theorem:

Theorem 1.4 (Computational Scheme for High Slices, Informal). *Let f be an $(n-k)$-slice with $k \leq \sqrt{n}$. Then if OWF exist, f can be realized by a computationally-secure secret-sharing scheme with share size $O(k^2\lambda\log n)$ (where λ is the security parameter). The running time of the sharing and reconstruction algorithms in the CSSS is* $\text{poly}(\binom{n}{k}, \lambda)$.[8]

Recall that by the previously-best scheme for k-slices based on OWFs has shares of size $\min\{kn, 2^{O(k)}\} \cdot \lambda \log n$ [AA18,ABF$^+$19,ABI$^+$23b]. Similarly to the computational scheme of Krawczyk and unlike perfect schemes, by Theorem 1.4 high slices now have smaller shares than low slices in CSSS.

Applications for Multislices. As applications of our perfect schemes for high slices, we present two schemes for $(a:b)$-multislices. The first one, optimized for the case where a and b are linear in n, solves an open question raised in [AN21], and has implications on the share size of general access structures. The second scheme is optimized for the regime where $a = n-k$ and $b = n$. The first scheme allows us to prove the following theorem:

Theorem 1.5 (Share Size of General Multislices). *For every $1 \leq a \leq b \leq n$, every $(a:b)$-multislice can be realized by a secret-sharing scheme with share size $\binom{n-a}{\geq n-b} \cdot 2^{o(n)}$.*

[8] As implied by [LS20,ABI$^+$23b], this running time is necessary for every CSSS for $(n-k)$-slices.

The share size of computational schemes for k-slices based on OWF

Slice height	k-slices	$(n-k)$-slices
Upper bounds	$\min\{kn, 2^{O(k)}\} \cdot \lambda \log n$ [AA18,ABF+19,ABI+23b]	$O(k^2 \lambda \log n)$ Theorem 4.1
Lower bounds	$\Omega(\log n)$ [ABI+23a]	2 [ABI+23a]

Fig. 2. The best-known upper and lower bounds on the share size for computational secret-sharing schemes for k-slice and $(n-k)$-slices based on OWF, for $k \leq n/2$. The lower bound [ABI+23a] does not require the OWF assumption; if we allow public information only a weaker bound of $\Omega(\log \log n)$ is proved in [ABI+23a].

This scheme closes the duality gap for multislices in the relevant regime. I.e., if we combine our scheme with that of [AN21] the share sizes of $(a : b)$-multislices and of their dual $(n-b : n-a, n)$-multislices are equal up to sub-exponential factors in n. We also prove the following theorem based on our second scheme for multislices, which is taylor-made for $(n-k : n)$-multislices for small k's.

Theorem 1.6 (Share Size of $(n-k:n)$-Multislices). *For every $k < \log n (\log \log n)^2$, every $(n-k : n)$-multislice can be realized by a secret-sharing scheme with share size $k^{5k} n 2^{\tilde{O}(\sqrt{k \log n})}$. For every $\log n (\log \log n)^2 \leq k \leq n/\log^2 n$, every $(n-k : n)$-multislice can be realized by a secret-sharing scheme with share size $2^{O(k)}$.*

For example, the share size for constant k's in this scheme is $n^{1+o(1)}$, and for $k = \log n$ it is $n^{O(\log \log n)}$, similarly to the $(n-k)$-slice schemes. When $\log n \leq k \leq \log n (\log \log n)^2$ the share size is $k^{5k+o(k)}$ and for larger values of k, the share size is $2^{O(k)}$. We note that our second construction also works for $(n-k)$-hypergraph functions, which are a specific subclass of $(n-k : n)$-multislices where the minterms are all of size $(n-k)$. I.e., a k-hypergraph acts the same as slices for inputs with weight $\leq k$, but outputs 1 on a heavier input y only if there exists a 1-input x of weight k such that $y \geq x$. [9] Hypergraph access structures were studied, e.g., in [AN21,Bei23]. We also present computational, linear, and multi-linear secret-sharing schemes for $(n-k : n)$-multislices (see Theorems 5.16 and 5.18).

Applications for Random Hypergraph Access Structures. Applebaum and Nir [AN21] studied the share size of "random hypergraphs". They showed that if a k-hypergraph f is chosen by drawing m_k minterms uniformly at random then with high probability the share size of f would be smaller than that of general k-hypergraphs. A result in the same spirit was proved in [BF20a] for small k's. More formally (yet still omitting some technical details), using multislices they proved that for every k and m_k the share size of hypergraphs generated

[9] We say that $y \geq x$ if in every coordinate $y_i \geq x_i$.

according to the above-mentioned procedure is $\sqrt{\binom{n}{k}} \cdot 2^{o(n)}$ with probability $1-2^{-\Omega(n)}$. The hardest random k-hypergraph in this case is when $k=n/2$ with shares of size $2^{n/2+o(n)}$. Applebaum and Nir also showed that balancing existing schemes for multislices with respect to duality (i.e., proving Theorem 1.5), would further improve this result. Hence, we prove the following corollary. We only state the improvement for the hardest random hypergraph, and refer the reader to [AN21, Theorem 6.2] for the general expression for every k which is somewhat involved.

Corollary 1.7 (Schemes for Random Hypergraphs). *For every $k \in [n]$, $m_k \leq \binom{n}{k}$, if a k-hypergraph is chosen by drawing m_k minterms of size k uniformly at random, then it can be realized with share size $2^{0.491n+o(n)}$ with probability $1-2^{-\Omega(n)}$.*

We note that general access structures can be easily realized given schemes for k-hypergraphs for $1 \leq k \leq n$; so this result may give hope for obtaining better schemes for general access structures with share size below $2^{0.5n}$.

1.3 Our Techniques

Perfect Schemes for High Slices and Duality. Our first scheme for $(n-k)$-slices relies on existing schemes for k-slices. The description below provides correctness and security for sets of size exactly $n-k$; correctness and security for sets of sizes below or above $n-k$ can be easily achieved with additional threshold schemes. To realize an $(n-k)$-slice access structure f, we start by generating shares to a k-slice function $\overline{f}$ determined by f, defined as $\overline{f}(x) = f(\overline{x})$ for every x of weight k. Then, each share sh_i of $\overline{f}$ is distributed with an $(n-k)$-out-of-$(n-1)$ threshold scheme among all of the parties except for the i-th one. Following this, let A be set of size $(n-k)$ whose characteristic vector x satisfies $f(x)=1$, i.e., $A=\{P_i : x_i = 1\}$. For each i such that $\overline{x}_i = 1$, i.e., $x_i = 0$, $P_i \notin A$ and the $n-k$ parties of A can recover sh_i. For P_i such that $P_i \in A$, the parties only hold $n-k-1$ shares of an $(n-k)$-threshold scheme, and thus will learn nothing about sh_i. In total, the parties of A can recover k shares of $\overline{f}$ that correspond to the coalition $\overline{A} = \{P_i : x_i = 0\}$, and since $\overline{f}(\overline{x}) = f(x) = 1$ this suffices to recover the secret. Similarly, unauthorized sets will recover $\overline{f}$-shares of unauthorized sets under f, and thus will learn nothing about the secret. This construction uses a "trick" introduced by Berkowitz [Ber82] of replacing a negated variable x_i with a threshold gate over all variables but the i'th one. Berkowitz used this idea to construct monotone formulas for k-slices from non-monotone formulas for k-slices, and Beimel, Kushilevitz and Nissim [BKN18] used it to construct secret-sharing schemes for slices from CDS protocols

We now return to the open problem discussed earlier: Can we bound the gap between the share sizes of slices and their duals? It is evident that our scheme solves this problem. Given better schemes for k-slices, we would be able to plug them into our construction and immediately get a better scheme for $(n-k)$-slices. This proves Theorem 1.3 stated above. Following this theorem, we make explicit some properties of our construction that we think may find future use by defining *duality compilers*:

Definition 1.8 (Duality Compilers). *Let $\mathcal{F}$ be a family of n-variable functions and let $\mathcal{F}^* \overset{def}{=} \{f^* : f \in \mathcal{F}\}$ (where f^* is the dual of f). A duality compiler for $\mathcal{F}$ is a transformation that takes as input secret-sharing schemes with share size $c_{\mathcal{F}}(n)$ for every function in a family $\mathcal{F}$ and a function $f^* \in \mathcal{F}^*$ and outputs a secret-sharing scheme realizing f^* with share size $c_{\mathcal{F}^*}(n)$. The goal in designing such compilers is to have a small blow-up ratio $c_{\mathcal{F}^*}(n)/c_{\mathcal{F}}(n)$.*

This definition expands the standard viewpoint on secret-sharing duality. Instead of examining specific functions and their duals, it shifts the focus to families of functions and their duals, and enables to draw new conclusions. We stress that in order to realize the function f, such duality compilers do not have to use a scheme for f^*, and they may, for example, use a scheme for a different function $f' \in \mathcal{F}$ or for a set of functions $S \subseteq \mathcal{F}$. Our construction provides a duality compiler for $(n-k)$-slices to their duals, i.e., k-slices, with blow-up n (where for every $(n-k)$-slice f, the construction uses a secret-sharing scheme for the k-slice $\overline{f}$).

Perfect Schemes for High Slices via CDS. Our second scheme for $(n-k)$-slices is based on *conditional disclosure of secrets* (CDS) protocols [GIKM00]. In a CDS protocol, there are k servers $\mathcal{S}_1, \dots, \mathcal{S}_k$, each holding a private input x_i, the secret s, and a common random string r, and a referee is holding the inputs $x_1, \dots, x_k$. Each server computes a message as a function of its input x_i, the secret s, and the common random string r (the message of each server is independent of the other inputs and is computed without seeing the other messages). Each server sends its message to the referee. We say that the CDS protocol realizes a function g if the referee can reconstruct s (from the k messages and the k inputs) if and only if $g(x_1, \dots, x_k) = 1$.

Given an $(n-k)$-slice, we will use a CDS protocol for a function $g_f : [n]^k \to \{0,1\}$ that encodes the way f behaves on inputs with weight $n-k$: On an input $(i_1, \dots, i_k)$ the function g_f outputs the same as f when given an input with 0's in the indices $(i_1, \dots, i_k)$ and 1's in all other indices. For example, on $i_1 = 2$, $i_2 = 4$ and $n = 5$ we define $g_f(2,4) = f(10101)$. Then, our goal is to distribute CDS messages generated according to g_f in a way that for every input x of weight $n-k$ the set of parties $A = \{P_i : x_i = 1\}$ will be able to recover k CDS messages, one from each server, that correspond to the input $\overline{x}$. I.e., in the previous example, $\{P_1, P_3, P_5\}$ should be able to reconstruct the message of the first server on input $i_1 = 2$ and the message of the second server on input $i_2 = 4$. Keeping the scheme based on slices in mind, a natural approach to do so would be to share every CDS message of the j-th server with the input i with an $(n-k)$-out-of-$(n-1)$ threshold scheme to all parties but the i-th one. However, this time a set of $n-k$ parties will be able to recover k messages of every CDS server, and in this case, the protocol does not guarantee any privacy. We will solve this issue by sharing the CDS messages in a more sophisticated way, inspired by the scheme for slices of [ABF$^+$19]. See examples and more technical details in Section 3.

Computational Schemes for High Slices. The starting point of our computational schemes for $(n-k)$-slices is to take the previously described perfect scheme based on CDS protocols and plug into it the computational CDS protocol of [ABI$^+$23b]. The share size in this implementation would be $O(nk\lambda \log n)$. While this is better than existing perfect schemes for high slices, we still need to save a multiplicative factor of n/k to prove Theorem 1.4. To do so, we notice that most of the shares dealt in the CDS-based scheme are of Shamir's threshold secret-sharing schemes with high thresholds. In an $(n'-t)$-out-of-n' Shamir scheme, $(n'-t-1)$ shares are independent random strings. Instead of dealing these random strings directly to the parties, we give each party only a (shorter) seed of a PRG, and the party generates its share from its seed. Our observation is that the same seed can be used for all schemes, which allows for further savings in the share size. In Shamir's scheme, $t+1$ of the shares are correlated with previous shares and we need to give them explicitly to the parties; with careful load balancing we can still get small shares as desired. For the full technical details, see Section 4.

Schemes for Multislices. Existing schemes for $(a:b)$-multislices are better when a and b are small. The scheme in [AN21] is aimed for the case where $a=\alpha n$, $b=\beta n$ for constants α,β, and it has smaller shares when these constants are small. A scheme in [BF20b] implicitly realizes $(0:k)$-multislices, and has huge shares for their dual $(n-k:n)$-multislices. We build schemes that complement the mentioned schemes and equalize the best-known share size for primal and dual multislices.

The constructions in [BF20b,AN21] both rely on formulas for multislices over CDS gates. By simple duality properties of formulas (Lemma 5.1), given such a formula F that computes a function f, if we replace in F every gate that computes a function g with a gate that computes the dual g^* of g, we will get a formula of the same size that computes f^*. Hence, in order to transform the known schemes for low multislices to schemes for high multislices it essentially suffices to realize the duals of CDS gates with small shares. The duals of CDS gates are functions that are somewhat contrived and hard to work with, and the key observation in our scheme is that the duals of k-server CDS gates can be replaced by $(n-k)$-slices. Hence, if we use our schemes for high slices we can realize high multislices with the same share size as low ones, employing the standard formula-to-scheme transformation (see Lemma 5.2 for a formal version).

1.4 Open Questions

Better Perfect Schemes for General Access Structures. In this work, we construct schemes for random hypergraphs. The obvious question to ask is whether these ideas (and shares of size $2^{\alpha n}$ for $\alpha < 1/2$) can be extended to schemes for worst-case hypergraphs, and from there to general access structures. The other side of this coin would be that random hypergraphs are easier for secret sharing than worst case ones.

Better Computational Schemes for Multislices from OWFs. When we construct $(n-k:n)$-multislices from $(n-k)$-slices, we follow a black-box transformation that is analogous to the construction of Robust CDS protocols (a generalization of CDS protocols defined in [ABNP20]) that adds a multiplicative factor of $k^{O(k)}$ to the share size. An improvement of this technique would lead to a reduction of the share size for multislices, and such an improvement may be easier to obtain taking advantage of one-way functions.

A Candidate for Duality-Separation. The best-known share size for k-slices with a constant $k \geq 2$ is $n^{o(1)}$, while that of their dual $(n-k)$-slices is now $n^{1+o(1)}$. It will be interesting to see whether this gap can be closed, or rather to prove that it is inherent. A possible path towards closing this gap may be to realize the dual of k-server CDS gates directly and more cheaply than our implementations of general $(n-k)$-slices.

Duality-compilers. Duality-compilers seem like a useful abstraction that may help obtain new bounds for secret sharing for families of functions. A natural next step would be to describe duality-compilers with a small blow-up for other families of functions. For example, the well-studied family of *graph access structures* where every minimal authorized coalition is of size 2, or its more general version of k-hypergraphs where every minimal authorized set is of size k.

1.5 Organization

In Section 3 we construct prefect schemes for high slices, proving Theorem 3.5 and Theorem 1.3. Thereafter, in Section 4 we present our computational scheme for high slices, proving Theorem 1.4. Finally, in Section 5 we show how our new schemes for slices help build better schemes for multislices, proving Theorem 1.5 and Theorem 1.6.

2 Preliminaries

2.1 Perfect Secret-Sharing Schemes

We define perfect secret-sharing scheme as given in [CK93,BC94]; in these schemes the security is information theoretic. Secret-sharing schemes with computational security will be defined in Section 2.3. For more information about this definition and secret-sharing in general, see [Bei11]. We start by defining an access structure, which is the collection of sets of parties that are authorized to reconstruct the secret. We describe an access structure by a monotone Boolean function.

Notation on Monotone Boolean Functions. The *weight* of an input $x \in \{0,1\}^n$, denoted $\mathrm{wt}(x)$, is the number of bits in x that are one, i.e., $\mathrm{wt}(x) = |\{i : x_i = 1\}|$. For two strings $x = (x_1, \dots, x_n)$,$y = (y_1, \dots, y_n) \in \{0,1\}^n$, we say that $x \leq y$ if $x_i \leq y_i$ for every $1 \leq i \leq n$. A function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $x \leq y$ implies $f(x) \leq f(y)$.

We will also consider partially defined functions, where $f(x) = *$ denotes that f is undefined on x. A partially defined function $f : \{0,1\}^n \to \{0,1,*\}$ is monotone if there does not exist $x, y \in \{0,1\}^n$ such that $x \leq y$, $f(x) = 1$, and $f(y) = 0$. A *minterm* of a monotone function f is a minimal input $x \in \{0,1\}^n$ such that $f(x) = 1$, i.e., for every $y \neq x$ if $y \leq x$ then $f(y) \in \{0,*\}$. A *maxterm* of a monotone function f is a maximal input $x \in \{0,1\}^n$ such that $f(x) = 0$, i.e., for every $y \neq x$ if $y \geq x$ then $f(y) \in \{1,*\}$.

Definition 2.1 (Access Structures). *An n-party access structure is a monotone function $f : \{0,1\}^n \to \{0,1,*\}$ such that $f(0^n) \neq 1$. Let $P = \{P_1, \dots, P_n\}$ be a set of parties; for an input $x = (x_1, \dots, x_n) \in \{0,1\}$, we define the set of parties that it represents as $I_x = \{P_i : x_i = 1\}$. For every $x \in \{0,1\}^n$, if $f(x) = 1$, then we say that I_x is authorized; if $f(x) = 0$, then we say that I_x is forbidden.*

A secret-sharing scheme is a randomized mapping $\Pi(s;r)$ whose input is a secret and a random string. 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 $\Pi(s;r) = (\mathsf{sh}_1, \dots, \mathsf{sh}_n)$, and privately communicating each share sh_j to party P_j. We require that any authorized set of parties can reconstruct the secret from its shares and any forbidden set cannot learn any information on the secret.

Definition 2.2 (Secret-Sharing Schemes). *A* secret-sharing scheme *Π with domain of secrets S, such that $|S| \geq 2$, is a mapping from $S \times R$, where R is some finite set called the set of random strings, to a tuple of n-sets $S_1 \times S_2 \times \cdots \times S_n$, where S_j is called the* domain of shares *of P_j. For an input $x \in \{0,1\}^n$, we denote $\Pi_x(s;r)$ as the restriction of $\Pi(s;r)$ to its I_x-entries, i.e., $(\mathsf{sh}_j)_{j:x_j=1}$.*

A secret-sharing scheme Π with domain of secrets S realizes an access structure $f : \{0,1\}^n \to \{0,1,*\}$ if the following two requirements hold:

Correctness. *For any input $x \in \{0,1\}^n$ such that $f(x) = 1$ there exists a* reconstruction function $\mathrm{Recon}_x : \prod_{\{i:x_i=1\}} S_i \to S$ *such that* $\mathrm{Recon}_x(\Pi_x(s;r)) = s$ *for every secret $s \in S$ and every random string $r \in R$.*

Security. *For any input $x \in \{0,1\}^n$ s.t. $f(x) = 0$ and every pair of secrets $s, s' \in S$, the distributions $\Pi_x(s;r)$ and $\Pi_x(s';r)$ are identical, where the distributions are over the choice of r from R at random with uniform distribution.*

Given a secret-sharing scheme Π, define the size *of the secret as* $\log|S|$, *the* share size *of party P_j as* $\log|S_j|$, *the* share size *as* $\max_{1 \leq j \leq n}\{\log|S_j|\}$, *the* total share size *as* $\sum_{j=1}^{n} \log|S_j|$, *and the* information ratio *as* $\frac{\max_{1 \leq j \leq n}\{\log|S_j|\}}{\log|S|}$.

By default, when we talk about the share size of secret-sharing schemes for an access structure, we consider schemes for one-bit secrets. Note that in Definition 2.2, there are no requirements for inputs x for which $f(x)$ is undefined, e.g., the parties in I_x can have partial information on the secret without being able to reconstruct it.

We next define multi-linear and linear secret-sharing schemes, which are schemes in which the mapping that the dealer uses to generate the shares is linear. Many of the known constructions of secret-sharing schemes are linear and multi-linear.

Definition 2.3 (Multi-Linear and Linear Secret-Sharing Schemes). *Let Π be a secret-sharing scheme with domain of secrets S. We say that Π is a* multi-linear *secret-sharing scheme over a finite field $\mathbb{F}$ if there are integers $\ell_d, \ell_r, \ell_1, \dots, \ell_n$ such that $S = \mathbb{F}^{\ell_d}$, $R = \mathbb{F}^{\ell_r}$, $S_1 = \mathbb{F}^{\ell_1}, \dots, S_n = \mathbb{F}^{\ell_n}$, and the mapping Π is a* linear *mapping over $\mathbb{F}$ from $\mathbb{F}^{\ell_d+\ell_r}$ to $\mathbb{F}^{\ell_1+\cdots+\ell_n}$. We say that a scheme is linear over $\mathbb{F}$ if $S = \mathbb{F}$ (i.e., when $\ell_d = 1$).*

Slice, Mutislice, and Hypergraph Access Structures. In this work we construct secret-sharing schemes for slices and for multislice access structures. A k-slice (also called uniform access structure) is an access structure where all sets of size smaller than k are forbidden, all sets of size larger than k are authorized, and sets of size k can be either forbidden or authorized. An $(a : b)$-multislice is an access structure where all sets of size smaller than a are forbidden, all sets of size larger than b are authorized, and sets of size between a and b can be either forbidden or authorized. A k-hypergraph access structure is an access structure whose minimal authorized sets are of size k, and it can have forbidden sets of size much larger than k.

Definition 2.4 (Slices, Multislices, and Hypergraphs). *Let k, n be integers such that $k \leq n$. A (k,n)-*slice *is a function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that if $\mathrm{wt}(x) < k$, then $f(x) = 0$ and if $\mathrm{wt}(x) > k$, then $f(x) = 1$. A* partially defined (k,n)-slice *is a function that is defined on all inputs of weight k and is undefined on all other inputs. When n is clear from the context, we write k slice instead of (k,n)-slice. Let a, b, n be integers such that $1 \leq a \leq b \leq n$. An (a,b)-*multislice *is a monotone function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that if $\mathrm{wt}(x) < a$, then $f(x) = 0$ and if $\mathrm{wt}(x) > b$, then $f(x) = 1$. A k-*hypergraph access structure *is a function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that all its minterms have weight exactly k.*

Note that a k-slice is a (k,k)-multislice and a k-hypergraph is a (k,n)-multislice.

Remark 2.5. To construct a secret-sharing scheme for a (fully-defined) k-slice, it suffices to construct a secret-sharing scheme for the partially defined function $f' : \{0,1\}^n \rightarrow \{0,1,*\}$, where $f'(x) = f(x)$ if $\mathrm{wt}(x) = k$ and $f'(x) = *$ otherwise. Given a secret-sharing scheme Π' realizing f', we construct a secret-sharing scheme Π with secret $s \in \{0,1\}$ realizing f as follows:

1. Share the secret s using a $(k+1)$-out-of-n secret-sharing scheme and give each party one share of this scheme.

2. Choose a random bit r_1 with uniform distribution and compute $r_2 = r_1 \oplus s$.
3. Share r_1 using a k-out-of-n secret-sharing scheme and give each party one share of this scheme.
4. Share r_2 using the secret-sharing scheme Π' and give each party its share of this scheme.

It can be verified that Π realizes the fully-defined k-slice f. The share size in Π is equal to the share size in Π' up to an additive term of $O(\log n)$. Thus, in this paper, we will realize partially defined slices.

2.2 Protocols for Conditional Disclosure of Secrets

We next define conditional disclosure of secrets (CDS) protocols, a useful cryptographic primitive introduced by Gertner et al. [GIKM00]. In particular, this primitive is used to construct secret-sharing schemes for general access structures, starting in the work of [LV18]. An informal presentation of CDS protocols appears in the introduction.

Definition 2.6 (Conditional Disclosure of Secrets (CDS) Protocols). *A* k-server CDS protocol $\mathcal{P}$*, with domain of secrets* S*, domain of common random strings* R*, and finite message domains* $M_1, \dots, M_k$*, consists of* k *encoding functions* $\textsc{Enc}_1, \dots, \textsc{Enc}_k$*, where* $\textsc{Enc}_i : X_i \times S \times R \to M_i$ *for every* $i \in [k]$*. For an input* $x = (x_1, \dots, x_k) \in X_1 \times \cdots \times X_k$*, secret* $s \in S$*, and randomness* $r \in R$*, we let* $\textsc{Enc}(x, s; r) = (\textsc{Enc}_1(x_1, s; r), \dots, \textsc{Enc}_k(x_k, s; r))$*.*

Let $g : X_1 \times \cdots \times X_k \to \{0, 1\}$ *be a* k*-input function. We say that* $\mathcal{P}$ *is a CDS protocol for* g *if it satisfies the following properties:*

Correctness. *There is a deterministic reconstruction function* $\textsc{Dec} : X_1 \times \cdots \times X_k \times M_1 \times \cdots \times M_k \to S$ *such that for every input* $x = (x_1, \dots, x_k) \in X_1 \times \cdots \times X_k$ *for which* $g(x_1, \dots, x_k) = 1$*, every secret* $s \in S$*, and every common random string* $r \in R$*, it holds that* $\textsc{Dec}(x, \textsc{Enc}(x, s; r)) = s$*.*

Security. *For every input* $x = (x_1, \dots, x_k) \in X_1 \times \cdots \times X_k$ *satisfying* $g(x_1, \dots, x_k) = 0$ *and every pair of secrets* $s, s' \in S$*, the distributions* $\textsc{Enc}(x, s; r)$ *and* $\textsc{Enc}(x, s'; r)$ *are equally distributed, where the probability distributions are over the choice of* r *from* R *with uniform distribution.*

The message size *of a CDS protocol* $\mathcal{P}$ *is defined as the size of the largest message sent by the servers, i.e.,* $\max_{1 \leq i \leq k} \log |M_i|$.

2.3 Computational Secret-Sharing Schemes and CDS Protocols

We quote the definition of *computational secret-sharing schemes (CSSS)* from [ABI$^+$23b]. In a $t(\lambda)$-secure CSSS the sharing and reconstruction are efficient, and no adversary running in time $t(\lambda)$ can learn non-negligible information about the secret from the shares of any unauthorized set of parties (where λ is the security parameter). When defining "efficiency" it is important to consider the way the access structure is represented. In this paper, we will mainly represent an access structure as a k-slice function, explicitly describing $f(x)$ for every input x of weight k. Nevertheless, in the definition of CSSS we use the abstract definition of a representation model.

Definition 2.7 (Representation Model [ABI$^+$23b]). *A* representation model *is a polynomial time computable function $U : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, where $U(\mathrm{Prog}, x)$ is referred to as the value returned by a "program"* Prog *on an input $x \in \{0,1\}^n$. We assume that each* Prog *specifies the input size n and* $|\mathrm{Prog}| \geq n$. *We say that* Prog *represents the function $f : \{0,1\}^n \to \{0,1\}$ in the representation model U if $U(\mathrm{Prog}, x) = f(x)$.*

Definition 2.8 (Comp. Secret-Sharing Schemes. (CSSS) [ABI$^+$23b]). *A CSSS for a representation model U consists of a pair of algorithms* CSSS = (CSSS.Share, CSSS.Recon) *with the following syntax.*

Sharing. $\mathrm{CSSS.S{\scriptsize HARE}}(1^\lambda, \mathrm{Prog}, s) \to (\mathsf{sh}_1, \ldots, \mathsf{sh}_n)$ *(where n denotes the input length of* Prog*) is a randomized poly-time algorithm that takes as input a security parameter λ, a program* Prog*, and a secret $s \in \{0,1\}$; it outputs n shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$, where sh_i, for $1 \leq i \leq n$, is the share of party P_i.*[10]

Reconstruction. $\mathrm{CSSS.R{\scriptsize ECON}}(\mathrm{Prog}, x, (\mathsf{sh}_i)_{i:x_i=1}) \to s$ *is a deterministic poly-time algorithm that takes as input a program* Prog*, an input $x \in \{0,1\}^n$ (where n denotes the input size of* Prog*), and shares of the parties in $I_x = \{P_i : x_i = 1\}$. The algorithm outputs a secret $s \in \{0,1\}$.*

We say that CSSS *is correct (with respect to U) if for every λ, s, program* Prog*, and input $x \in \{0,1\}^n$ such that $U(\mathrm{Prog}, x) = 1$ (where n denotes the input length of* Prog*), the process of invoking*

$$\mathrm{CSSS.S{\scriptsize HARE}}(1^\lambda, \mathrm{Prog}, s) \to (\mathsf{sh}_1, \ldots, \mathsf{sh}_n)$$

and then invoking $\mathrm{CSSS.R{\scriptsize ECON}}(\mathrm{Prog}, x, (\mathsf{sh}_i)_{i:x_i=1})$ *always returns s.*

To define the security of CSSS *we consider the following game between a non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and a challenger:*

1. *The adversary $\mathcal{A}$ on input 1^λ chooses* Prog *and an input $x \in \{0,1\}^n$ such that $U(\mathrm{Prog}, x) = 0$ (where n is the input size of* Prog*) and sends them to the challenger.*
2. *The challenger chooses a secret $s \leftarrow_U \{0,1\}$ uniformly at random. It computes $(\mathsf{sh}_1, \ldots, \mathsf{sh}_n) \leftarrow \mathrm{CSSS.S{\scriptsize HARE}}(1^\lambda, \mathrm{Prog}, s)$ and sends $(\mathsf{sh}_i)_{x_i=1}$ to the adversary.*
3. *The adversary outputs a bit s'.*

The adversary wins the game if $s' = s$. We say that CSSS *is $t(\lambda)$-secure if for every non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and sufficiently large λ, the probability*

[10] In [ABI$^+$23b], the scheme also returns public information sh_0 given to all parties (or published in the cloud); in this work we do not use this public information.

that $\mathcal{A}$ wins is at most $1/2+1/t(\lambda)$. By default, we require $t(\lambda)$-security for every polynomial $t(\cdot)$. In any case, we always assume that $1/t(\lambda)$ is negligible.[11]

A computational CDS (CCDS) protocol is defined similarly to CSSS.

Definition 2.9 (Computational CDS Protocols). *A computational CDS protocol for a representation model U consists of a pair of algorithms $\mathrm{CCDS} = (\mathrm{CCDS}.\textsc{Enc}, \mathrm{CCDS}.\textsc{Dec})$ with the following syntax.*

Encoding. $\mathrm{CCDS}.\textsc{Enc}(1^\lambda, \mathrm{Prog}, i, x_i, s; r) \to m_i$ *is a randomized poly-time algorithm that takes as input a security parameter λ, a program* Prog*, an index $1 \leq i \leq k$, an input $x_i \in \{0,1\}^\ell$, a secret $s \in \{0,1\}$, and a common random string r (where $k\ell$ denotes the input length of* Prog*). It outputs a message m_i.*

Decoding. $\mathrm{CCDS}.\textsc{Dec}(\mathrm{Prog}, m_1, \dots, m_k, x_1, \dots, x_k) \to s$ *is a deterministic poly-time algorithm that takes as input a program* Prog*, k messages, and k inputs, each one of length ℓ (where $k\ell$ denotes the input size of* Prog*). The algorithm outputs a secret $s \in \{0,1\}$.*

We say that CCDS *is correct (with respect to U) if for every λ, s, program* Prog*, inputs $x_1, \dots, x_k \in \{0,1\}^\ell$ such that $U(\mathrm{Prog}, (x_1, \dots, x_k)) = 1$ (where $k\ell$ denotes the input length of* Prog*), and common random string r, the process of invoking*

$$\mathrm{CCDS}.\textsc{Enc}(1^\lambda, \mathrm{Prog}, i, x_i, s; r) \to m_i \qquad \textit{for every } 1 \leq i \leq k,$$

and then invoking $\mathrm{CCDS}.\textsc{Dec}(\mathrm{Prog}, m_1, \dots, m_k, x_1, \dots, m_k)$ always returns s.

To define the security of CCDS *we consider the following game between a non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and a challenger:*

1. *The adversary $\mathcal{A}$ on input 1^λ chooses* Prog *and inputs $x_1, \dots, x_k \in \{0,1\}^\ell$ such that $U(\mathrm{Prog}, (x_1, \dots, x_k)) = 0$ (where $k\ell$ is the input size of* Prog*) and sends them to the challenger.*
2. *The challenger chooses a secret $s \leftarrow_U \{0,1\}$ uniformly at random and samples a common random string r. For every $1 \leq i \leq k$, the challenger computes $m_i \leftarrow \mathrm{CCDS}.\textsc{Dec}(1^\lambda, \mathrm{Prog}, i, x_i, s; r)$; it then sends $m_1, \dots, m_k$ to the adversary.*
3. *The adversary outputs a bit s'.*

The adversary wins the game if $s' = s$. We say that CCDS *is $t(\lambda)$-secure if for every non-uniform $t(\lambda)$-time adversary $\mathcal{A}$ and sufficiently large λ, the probability that $\mathcal{A}$ wins is at most $1/2+1/t(\lambda)$. By default, we require $t(\lambda)$-security for every polynomial $t(\cdot)$. In any case, we always assume that $1/t(\lambda)$ is negligible.*

[11] A function $\varepsilon(\lambda)$ is negligible if for every positive polynomial (λ) there exits λ_0 such that $\varepsilon(\lambda) \leq 1/p(\lambda)$ for every $\lambda > \lambda_0$. Our results remain valid also when $t(\lambda) \geq \lambda$, as in [ABI$^+$23b]; for simplicity of our notations we prefer to only consider negligible functions.

Theorem 2.10 ([ABI$^+$23b]). *Assuming $t(\lambda)$-secure one-way functions exist, for all k-input functions $g : (\{0,1\}^\ell)^k \to \{0,1\}$, represented by truth tables of size $N = 2^{\ell k}$, there exists a* $\mathrm{poly}(t(\lambda))$*-secure CCDS protocol with message size $O(\lambda\ell)$. The running time of the encoding and decoding algorithms is $O(2^{\ell k}\lambda)$.*

Remark 2.11. Applebaum et al. [ABI$^+$23c, Theorem 5.2] state their result as a construction of a CSSS for k-partite access structures in which the share size is $\lambda + O(1)$. By a simple transformation, this implies a k-server CCDS protocol with share size $\lambda + O(1)$ for 1-bit inputs. To get a k-server CCDS protocol for ℓ-bit inputs, we start with a $k\ell$-server CCDS protocol for 1-bit inputs, partition the servers to k sets of size ℓ, and simulate each set by a server of the k-server CCDS protocol; this increases the message size by a multiplicative factor of ℓ.

3 Perfect Secret-Sharing Schemes for $(n-k)$-Slices

We provide two new constructions of perfect secret-sharing schemes for $(n-k)$-slice functions. The first one is based on schemes for k-slices, and the second one on k-server CDS protocols. By the current state of the art of CDS protocols and secret-sharing schemes for slices, the second scheme is more efficient by a factor of n. For small k's it has shares of size $kn^{1+o(1)}$, compared to shares of size $kn^{2+o(1)}$ for the first simpler scheme. However, if more efficient schemes for slices will be constructed, the first scheme may become the leading one.

3.1 Construction from Schemes for k-Slices

In this section, we prove Theorem 3.1, which is a reformulation of Theorem 1.3. We describe a simple scheme for $(n-k)$-slices based on a scheme for k-slices. Specifically, we will show how to realize an $(n-k)$-slice f given a scheme for the partially-defined k-slice $\overline{f}$, where $\overline{f}(x) = f(\overline{x})$ for every input x of weight k and undefined for other inputs.[12]

The secret: An element $s \in \{0,1\}$.
The scheme:

1. Share the secret s with a secret-sharing scheme $\overline{\Pi}$ realizing $\overline{f}$; denote by $\mathsf{sh}_1, \dots, \mathsf{sh}_n$ the resulting shares.
2. For every $t \in [n]$, share sh_t with Shamir's $(n-k)$-out-of-$(n-1)$ secret-sharing scheme, and distribute one share to each party in $\{P_1, \dots, P_n\} \setminus \{P_t\}$.

The share of each P_j: A share of each sh_t for every $t \neq j$.

Fig. 3. A secret-sharing scheme realizing a partially defined $(n-k)$-slice f using a scheme $\overline{\Pi}$ for the partially defined k-slice $\overline{f}$.

[12] For an input $x = (x_1, \dots, x_n) \in \{0,1\}^n$, we define $\overline{x} = (\overline{x}_1, \dots, \overline{x}_n)$.

Theorem 3.1. *Let $f : \{0,1\}^n \rightarrow bit$ be an $(n-k)$-slice. If there is a secret-sharing scheme for the partially defined k-slice $\overline{f}$ with share size $c_{\text{slice}}(k,n)$, then there is a secret-sharing scheme realizing the slice f with share size $O(n \cdot \max\{\log n, c_{\text{slice}}(k,n)\})$.*

Proof. By Remark 2.5, it suffices to realize partially-defined slice functions, only defined on inputs of weight $n-k$. The scheme for such a function f is described in Fig. 3. We next prove the correctness and security of the scheme, only considering inputs of weight $n-k$.

For correctness, if $f(x) = 1$ then by definition $\overline{f}(\overline{x}) = f(x) = 1$, and hence the shares $\{\mathsf{sh}_i : \overline{x}_i = 1\} = \{\mathsf{sh}_i : x_i = 0\}$ of $\overline{\Pi}$ reveal the secret. For every i such that $x_i = 0$, the parties of I_x can compute every sh_i generated in Step 1 by combining their corresponding $n-k$ shares in the threshold sharing of sh_i dealt in Step 2 (since $P_i \notin I_x$), and thus can recover the secret s.

For security, if $f(x) = 0$ then by definition $\overline{f}(\overline{x}) = f(x) = 0$. For every i such that $x_i = 1$, the parties in I_x only hold the shares of $I_x \setminus \{P_i\}$, i.e., they hold $n-k-1$ shares in a secret-sharing scheme with threshold $n-k$ and these shares are uniformly distributed. Thus, the parties in I_x can only obtain the shares $\{\mathsf{sh}_i : x_i = 0\} = \{\mathsf{sh}_i : \overline{x}_i = 1\}$ of $\overline{\Pi}$ from the shares generated in Step 1 for the access structure $\overline{f}$. These are shares of the set $I_{\overline{x}}$ which is an unauthorized set of $\overline{f}$; therefore they reveal no information on s.

The share of each party consists of $n-1$ shares of shares of $\overline{\Pi}$; each share in a Shamir threshold scheme has size $O(\max\{\log n, c_{\text{slice}}(n,k)\}$. □

Using the k-slice secret-sharing scheme of [ABF+19], which has share size $kn \cdot 2^{\tilde{O}(\sqrt{k \log n})}$, in Theorem 3.1, results in a scheme for $(n-k)$-slices with share size $n^2 k \cdot 2^{\tilde{O}(\sqrt{k \log n})}$. In the following section we prove Theorem 3.5 by presenting a better scheme.

3.2 Construction from k-Server CDS Protocols

We now preset the second construction for $(n-k)$-slices. The structure of this construction is similar to the construction from [ABF+19] of secret-sharing schemes for k-slices from k-server CDS protocols. For that, we need to define the following functions; in these functions we encodes an input of weight $(n-k)$ by the k indices in which the input is 0.

Definition 3.2 (The Function g_f). *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be an $(n-k)$-slice. For a sequence $j_1, \dots, j_k$ of k distinct numbers in $[n]$ we define an input $X^{j_1,\dots,j_k} = (x_1, \dots, x_n)$ as $x_{j_1} = x_{j_2} = \cdots = x_{j_k} = 0$ and all other bits of x are 1; the weight of $X^{j_1,\dots,j_k}$ is exactly $n-k$. We define the k-input function $g_f : [n]^k \rightarrow \{0,1\}$, where $g_f(j_1, \dots, j_k) = 1$ if and only if $1 \leq j_1 < \cdots < j_k \leq n$ and $f(X^{j_1,\dots,j_k}) = 1$.*

Example 3.3. For the sequence $(2,3)$, the input $X^{2,3}$ is 1001^{n-3}. Let f be the $(n-2)$-slice function, where $f(x) = 1$ for an input $x = (x_1, \dots, x_n)$ of weight

$n-2$ if and only if there is an index $1 \leq j \leq n-1$ such that $x_j = x_{j+1} = 0$. In this case $g_f(j_1, j_2) = 1$ if and only if $j_2 = j_1 + 1$. E.g., for $n = 5$,

$$g_f(2,3) = f(X^{2,3}) = f(10011) = 1 \text{ and } g_f(2,4) = f(X^{2,4}) = f(10101) = 0.$$

In Fig. 4, we describe the secret-sharing scheme realizing a partially defined $(n-k)$-slice f using a k-server CDS protocol for g_f. We next describe the ideas of the scheme, considering an $(n-2)$-slice f. We execute a 2-server CDS protocol for g_f; let $m_{i,j}$ be the message of server $\mathcal{S}_i$ in the 2-server CDS protocol with input $j \in [n]$. Consider an input x of weight $n-2$ such that $f(x) = 1$ and let $j_1 < j_2$ be the indices such that $x_{j_1} = x_{j_2} = 0$, i.e., $x = X^{j_1,j_2}$. Thus, $g_f(j_1, j_2) = 1$ and the secret can be reconstructed from m_{1,j_1}, m_{2,j_2}. We can try and apply the same strategy as in the scheme described in Fig. 3, that is, sharing each message $m_{i,j}$ in an $(n-2)$-out-of-$(n-1)$ secret-sharing scheme and give the shares to all parties except for P_j. In this case, the parties in I_x can reconstruct m_{1,j_1}, m_{2,j_2} and reconstruct the secret. However, when $f(x = X^{j_1,j_2}) = 0$ the parties in I_x can also reconstruct m_{1,j_2}, m_{2,j_1} (as, for example, the shares of m_{1,j_2} are given to all parties except for P_{j_2}). In this case, the parties in I_x can reconstruct two messages of the first server and there are no security guarantees from the CDS protocol.[13]

We need to ensure that the parties in I_x can only reconstruct the message $m_{1,j}$, where j is the smallest index such that $x_j = 0$. In this case $x_1 = \cdots = x_{j-1} = 1$ and all the $n-j$ bits $x_{j+1}, \ldots, x_n$ are 1 except for exactly one bit. Thus, for every j we share $m_{1,j}$ in a 2-out-of-2 secret-sharing scheme. We share the first share in a $(j-1)$-out-of-$(j-1)$ secret-sharing scheme and give the shares to the first $j-1$ parties. Similarly, we share the second share in a $(n-j-1)$-out-of-$(n-j)$ secret-sharing scheme and give the shares to the last $n-j$ parties. We treat $m_{2,j}$ symmetrically. Some technical details arise in the first and last indices. For example, $j_2 \geq 1$, so we do not need to share $m_{2,1}$. As another example, if $j_1 = 1$, there are no parties with index smaller than 1 and we share $m_{1,1}$ in a $(n-k)$-out-of-$(n-1)$ scheme (without sharing it in a 2-out-of-2 scheme).

The scheme for $(n-k)$-slice functions generalizes this idea, where each message $m_{i,j}$ is shared using a 2-out-of-2 secret-sharing scheme, the first and second shares are shared among the first $j-1$ parties and last $n-j$ parties respectively with appropriate thresholds.

Lemma 3.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be an $(n-k)$-slice. If there is a k-server CDS protocol for $g_f : [n]^k \to \{0,1\}$ with message size $c_{\text{cds}}(k,n)$, then there is a secret-sharing scheme realizing f with share size $O(k \cdot n \cdot \max\{\log n, c_{\text{cds}}(k,n))\}$.*

Proof. By Remark 2.5, it suffices to realize partially-defined slice functions, only defined on inputs of weight $n-k$. The scheme for such a function f is described in Fig. 4. We next prove the correctness security of the scheme, only considering

[13] This problem can be solved by using robust CDS protocols (as defined in [ABNP20]); however, the known robust CDS protocols have very large message size. We use an idea of [ABF$^+$19] to solve this problem.

The secret: An element $s \in \{0,1\}$.
Auxiliary protocol: Let $\mathcal{P}$ be a k-server CDS protocol with message size $c = c_{\mathrm{cds}}(k,n)$ for the function $g_f : [n]^k \to \{0,1\}$ (defined in Definition 3.2).

The scheme:

1. Choose a common random string r as chosen in the k-server CDS protocol $\mathcal{P}$ and execute $\mathcal{P}$ with the secret s and the common random string r. For every $i \in [k], j \in [n]$, let $m_{i,j} \in \{0,1\}^c$ be the message of server $\mathcal{S}_i$ in $\mathcal{P}$ with input $j \in [n]$, i.e., $m_{i,j} = \textsc{Enc}_i(j, s; r)$.
2. For $i+1 \leq j \leq n-k+i-1$, share $m_{i,j}$ in a 2-out-of-2 secret-sharing scheme, that is, choose $m^1_{i,j}$ with uniform distribution from $\{0,1\}^c$ and compute $m^2_{i,j} = m_{i,j} \oplus m^1_{i,j}$.
3. Let $m^2_{i,i} = m_{i,i}$ and $m^1_{i,n-k+i} = m_{i,n-k+i}$ for every $i \in [k]$.
4. For every $i \in [k]$ and $i+1 \leq j \leq n-k+i$, share the string $m^1_{i,j}$ using the $(j-i)$-out-of-$(j-1)$ secret-sharing scheme among the first $j-1$ parties.
5. For every $i \in [k]$ and every $i \leq j \leq n-k+i-1$, share the string $m^2_{i,j}$ using an $(n-k-j+i)$-out-of-$(n-j)$ secret-sharing scheme among the last $n-j$ parties.

Fig. 4. A secret-sharing scheme realizing a partially defined $(n-k)$-slice f using a k-server CDS protocol $\mathcal{P}$ for g_f.

inputs of weight $n-k$. That is, we consider an input x such that $x_{j_i} = \cdots = x_{j_k} = 0$ for some indices $1 \leq j_i \leq \cdots \leq j_k \leq n$ and all other bits in x are 1, i.e., $x = X^{j_i,\cdots,j_k}$.

Assume that $f(x) = 1$. We next explain how the parties in I_x can recover m_{i,j_i} for every $i \in [k]$. First observe that $i \leq j_i \leq n-k+i$ (since there are $i-1$ bits in x that are zero before x_{j_i} and there are $k-i$ bits that are zero after x_{j_i}). If $i = j_i$, then $x_1 = x_2 = \cdots = x_i = 0$ and I_x has $(n-i)-(k-i) = n-k$ parties with index greater than $j_i = i$, i.e., the parties in I_x can recover $m^2_{i,j_i} = m_{i,j_i}$, which is shared via a secret-sharing scheme with threshold $n-k-j_i+i = n-k$. Analogously, if $j_i = n-k+i$, then $x_{n-k+i} = x_{n-k+i+1} = \cdots = x_n = 0$ and I_x has $(j_i-1)-(i-1) = n-k+i-1-i+1 = n-k$ parties with index smaller than $j_i = n-k+i$, i.e., the parties in I_x can recover $m^1_{i,j_i} = m_{i,j_i}$. Now consider the case that $i+1 \leq j_i \leq n-k+i-1$. The subset I_x has $j_i - i$ parties with index smaller than j_i and $n-j_i-(k-i)$ parties with index greater than j_i. So the subset I_x can recover both m^1_{i,j_i} and m^2_{i,j_i} and so can compute m_{i,j_i}. As $g_f(j_1,\ldots,j_k) = 1$ and the parties in I_x can recover the messages $m_{1,j_1},\ldots,m_{k,j_k}$, they can recover the secret.

Assume that $f(x) = 0$. In this case $g_f(j_1,\ldots,j_k) = 0$, hence the parties in I_x cannot obtain any information on s from the messages $m_{i,j_1},\ldots,m_{k,j_k}$ they can recover. We next claim that the parties in I_x have no information on any message $m_{i,j}$, where $j \neq j_i$. Since $j \neq j_i$, the number of bits that are zero in x in the first j bits of x is not i, i.e., either there are at least i bits that are zero among the first $j-1$ bits of x or there are at least $k-i$ bits that are zero among the last $n-j$ bits of x (since exactly k bits of x are zero). In the former case, the parties in I_x hold at most $j-1-i$ shares in a secret-sharing scheme of $m^1_{i,j}$ with threshold $j-i$, hence they have no information on $m^1_{i,j}$, thus, they have no information on $m_{i,j}$. In the latter case, the parties in I_x hold at most $(n-j)-(k-i)=n-k-j$ shares in a secret-sharing scheme of $m^2_{i,j}$ with threshold $n-k-j+1$, hence they have no information on $m^2_{i,j}$, thus, they have no information on $m_{i,j}$. As each $m_{i,j}$ is shared independently, the parties in I_x gain no information from the sharing of all the messages $(m_{i,j})_{i\in[k],j\neq i_j}$. To conclude, the set I_x only obtains the messages $m_{i,j_1},\dots,m_{k,j_k}$, which by the security of the CDS protocol give no information on the secret.

The share of each party is composed of $O(nk)$ shares in Shamir's threshold secret-sharing schemes with $O(n)$ parties. Thus, the share size is $O(nk \cdot \max\{\log n, c_{\text{cds}}(k,n)\})$. □

Theorem 3.5 (Perfect Schemes for High Slices, Theorem 1.2 Restated). *Let $k \leq n/2$ be positive integers. For every $(n-k)$-slice function f, there exists a secret-sharing scheme realizing f with share size*

$$kn \cdot 2^{O(\sqrt{k\log n})\log(k\log n)} = kn \cdot 2^{\tilde{O}(\sqrt{k\log n})}.$$

Proof. By [LVW18], there is a k-server CDS protocol for functions $g:[n]^k \rightarrow \{0,1\}$ with message size $2^{O(\sqrt{k\log n}\log(k\log n))}$. Using this protocol in Lemma 3.4, we get for every $(n-k)$-slice a secret-sharing scheme with the share size stated in the theorem. □

4 Computationally-Secure Schemes for $(n-k)$-Slices

In this section, we construct CSSSs for $(n-k)$-slices. The first observation is that for $k \leq \sqrt{n}$ we can use the CCDS protocol of [ABI$^+$23b] (see Theorem 2.10) in the scheme of Lemma 3.4 and obtain a secret-sharing scheme for $(n-k)$-slices with share size $O(k\lambda n\log n)$; this is a slight improvement compared to the perfect scheme we constructed. Similarly, for $k \leq \sqrt{n}$ we can obtain a CSSS realizing k-slices with share size $O(\min\left\{2^k, nk\right\}\cdot k\lambda\log n)$ by plugging the CCDS protocol of [ABI$^+$23b] in the schemes of [AA18,ABF$^+$19].[14]

Our goal is to save a factor of n/k in the share size of CSSS realizing $(n-k)$-slices compared to the above-mentioned CSSS for $(n-k)$-slices. Recall that in our scheme described in Lemma 3.4, the share of each party contains $O(kn)$ shares in threshold secret-sharing schemes with secrets of size $c_{\text{ccds}}(\lambda,k,n)=O(\lambda\log n)$. We show how to realize these secret-sharing schemes such that the share size of each party in the $O(kn)$ threshold schemes is $O(k^2\lambda\log n)$. The idea is to give each party a seed of a pseudorandom generator (PRG) that is expanded to a pseudorandom string containing the $O(kn)$ shares. The obstacle is that the n

[14] The proofs of the constructions of [AA18,ABF$^+$19] and Lemma 3.4 are when the CDS protocol is perfect; however, they can be updated to the computational setting, similarly to the proof of Claim 4.2.

shares are correlated. We use the fact that in the t-out-of-n secret-sharing of Shamir [Sha79] the first $t-1$ shares are uniformly distributed and independent. While the schemes described in the previous paragraph are relatively simple – they take a formula for f and realize some of its gates with computational schemes instead of perfect ones, our scheme for $(n-k)$-slices treats the perfect scheme as "white box", replacing the shares that are random by pseudorandom strings. This methodology is not new; however, the way we utilize it is new.

In Fig. 6, we present the sharing in Shamir's scheme making this fact explicit; in our presentation, the dealer gives a random element to a set A of $t-1$ parties and then picks a polynomial Q that interpolates the $t-1$ shares of A and the secret to generate the shares for the rest of the parties B. Note that in this scheme the polynomial Q is a uniformly distributed polynomial of degree at most $t-1$ such that $Q(0)=s$, that is, the sharing is exactly as in the more common description of Shamir's secret-sharing scheme. This is similar to the systematic encoding of Reed-Solomon codes. We use Procedure Interpolate, described in Fig. 5, to compute the polynomial. We will use this procedure in our scheme for $(n-k)$-slices; the above sets A and B will be carefully chosen to minimize the share size of each party.

Procedure Interpolate$(t, A, B, s, (\mathsf{sh}_i)_{P_i \in A})$**:**
Parameters: A threshold t, a set of $t-1$ parties A, a set of parties B disjoint from A, a secret s, and the shares $(\mathsf{sh}_i)_{P_i \in A}$ of the parties in A.

1. The dealer computes the unique polynomial Q of degree at most $t-1$ in $\mathbb{F}_p[x]$ that satisfies $Q(0)=s$ and $Q(i)=\mathsf{sh}_i$ for every $P_i \in A$.
2. The dealer computes $\mathsf{sh}_i = Q(i)$ for every $P_i \in B$ and returns $(\mathsf{sh}_i)_{P_i \in B}$.

Fig. 5. A description of Procedure Interpolate that, given shares of $t-1$ parties and a secret, uses interpolation to find the polynomial that passes via these points and computes the other shares using this polynomial.

We can change Shamir's scheme as described in Fig. 6, giving each party in A an independent seed w_i of a pseudorandom generator (PRG), and for $P_i \in A$, the party P_i and the dealer compute $\mathsf{sh}_i = \mathrm{PRG}(w_i)$. The dealer also computes shares $(\mathsf{sh}_i)_{P_i \in B}$ from $(\mathsf{sh}_i)_{P_i \in A}$ and the secret (using Procedure Interpolate) and gives these shares to the parties in B. The total share size in this scheme is $O(t\lambda + (n-t)\log p)$ (where Shamir's scheme is executed over $\mathbb{F}_p$ for a prime $p > n$). [15] In a single execution of this scheme, using PRGs reduces the total share size when $n-t$ is small and the length of the secret is bigger than the security parameter (i.e., it avoids the lower bound of the length of the secret

[15] The same results also hold for $\mathbb{F}_q$, where $q > n$ is a prime power. For the sake of simplicity, we restrict the presentation to the case that q is a prime number.

The secret: An element $s \in \mathbb{F}_p$, where $p > n$ is a prime.
Public parameters: A set $A \subset P = \{P_1, \ldots, P_n\}$ such that $|A| = t-1$.
The scheme:

1. For every $P_i \in A$, choose sh_i independently with uniform distribution from $\mathbb{F}_p$.
2. Interpolate$(t, A, P \setminus A, s, (\mathsf{sh}_i)_{P_i \in A})$.

The shares: $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$.

Fig. 6. Shamir's t-out-of-n secret-sharing scheme with a systematic choice of the polynomial.

that holds for perfect secret-sharing schemes [KGH83]). Note that Krawczyk's construction [Kra94] gives a smaller share size for this case.[16]

When using the scheme many times with different secrets, the saving is more dramatic – the dealer can give each of the first $t-1$ parties only one seed w_i, which will be expanded to the shares of the party in all the schemes, that is, the share size of these parties is $O(\lambda)$. This can be done even when the thresholds in the various secret-sharing schemes are not the same, and for different sets A. Specifically, in our secret-sharing scheme for $(n-k)$-slices we execute $O(nk)$ threshold t-out-of-n' secret-sharing schemes with various t, n' such that $n'-t \leq k$ and $\log p = O(\lambda \log n)$ (this is the length of the messages in the CCDS protocol, which we need to share with Shamir's scheme). The total share size in these executions is $O(n\lambda + k^2\lambda n \log n)$, i.e., one seed per party and k "extra" shares for each of the $O(nk)$ executions of the threshold scheme, each "extra" share is of length $O(\lambda \log n)$. We can balance the share size in these $O(nk)$ executions by giving each of the n parties "extra" shares only in $O(k^2)$ schemes; this results in share size $O(k^2\lambda \log n)$ per party.

The following Theorem 4.1 is a formal statement of Theorem 1.4 from the introduction; in the formal statement of the theorem we deal with the family of slice functions rather than a specific function (as required by Definition 2.8 – the definition of CSSS).

Theorem 4.1. *Let $k : \mathbb{N} \rightarrow \mathbb{N}$ be a function such that $2 \leq k(n) \leq \sqrt{n}$ for every $n \in \mathbb{N}$. If $t(\lambda)$-secure one-way functions exist for some negligible function $1/t(\lambda)$, then there exists a* $\mathrm{poly}(t(\lambda))$*-secure CSSS for $(n-k(n))$-slice functions represented as a truth table of size $\binom{n}{k(n)}$ with share size $O(k^2\lambda \log n)$. The running time of the sharing and reconstruction algorithms of the CSSS is* $\tilde{O}(n^k) \cdot \mathrm{poly}(\lambda) = \mathrm{poly}(\binom{n}{k(n)}, \lambda)$.

[16] We cannot use Krawczyk's construction as we have a few schemes with different thresholds and different sets of parties.

Input: A secret $s \in \{0,1\}$ and an $(n-k)$-slice access structure f described by a truth table of size $\binom{n}{k}$.

Auxiliary tools:

- A k-server CCDS protocol CCDS $= (\text{CCDS.}\textsc{Enc}, \text{CCDS.}\textsc{Dec})$ with message size $c = c_{\text{ccds}}(\lambda, k, n)$ for $g_f : [n]^k \rightarrow \{0,1\}$ (defined in Definition 3.2).
- A pseudorandom generator PRG that gets as an input a seed w of length λ and outputs a string of length knc; denote $\text{PRG}(w) = (\text{PRG}_{i,j}(w))_{i\in[k],j\in[n]}$, where $\text{PRG}_{i,j}(w) \in \{0,1\}^c$.
- Procedure Interpolate from Fig. 5 over a field $\mathbb{F}_p$, where $\log p \approx c$ is the message size of the CCDS protocol.

The scheme:

1. For every $\ell \in [n]$, sample with uniform distribution a seed $w_\ell \in \{0,1\}^\lambda$ for the party P_ℓ and let $y_\ell^{i,j} \leftarrow \text{PRG}_{i,j}(w_\ell)$ for every $i \in [k], j \in [n]$.
2. Choose a common random string r as chosen in the k-server CDS protocol CCDS and execute CCDS with the secret s and the common random string r. For every $i \in [k], j \in [n]$, let $m_{i,j} \in \{0,1\}^c$ be the message $m_{i,j} \leftarrow \text{CCDS.}\textsc{Enc}(1^\lambda, g_f, i, x_i = j, s; r)$.
3. For every $i \in [k]$ and $i+1 \leq j \leq n-k+i-1$, share $m_{i,j}$ in a 2-out-of-2 secret-sharing scheme, that is, choose $m^1_{i,j}$ with uniform distribution from $\{0,1\}^c$ and compute $m^2_{i,j} = m_{i,j} \oplus m^1_{i,j}$.
4. For every $i \in [k]$, let $m^2_{i,i} = m_{i,i}$ and $m^1_{i,n-k+i} = m_{i,n-k+i}$.
5. For every $i \in [k]$ and $i+1 \leq j \leq n-k+i$, share the string $m^1_{i,j}$ with a $(j-i)$-out-of-$(j-1)$ secret-sharing scheme among the first $j-1$ parties [a]: Let $A = \{P_1, \dots, P_{j-i-1}\}, B = \{P_{j-i}, \dots, P_{j-1}\}$, $\mathsf{sh}_\ell^{i,j,1} \leftarrow y_\ell^{i,j}$ for $1 \leq \ell \leq j-i-1$, $t = j-i$, and
$$\left(\mathsf{sh}_\ell^{i,j,1}\right)_{\ell=j-i}^{j-1} \leftarrow \text{Interpolate}\left(t, A, B, s = m^1_{i,j}, (\mathsf{sh}_\ell^{i,j,1})_{\ell=1}^{j-i-1}\right).$$
6. For every $i \in [k]$ and every $i \leq j \leq n-k+i-1$, share the string $m^2_{i,j}$ with an $(n-k-j+i)$-out-of-$(n-j)$ secret-sharing scheme among the last $n-j$ parties: Let $A = \{P_{j+k-i+2}, \dots, P_n\}, B = \{P_{j+1}, \dots, P_{j+k-i+1}\}$, $\mathsf{sh}_\ell^{i,j,2} \leftarrow y_\ell^{i,j}$ for $j+k-i+2 \leq \ell \leq n$, $t = n-k-j+1$, and
$$\left(\mathsf{sh}_\ell^{i,j,2}\right)_{\ell=j+1}^{j+k-i+1} \leftarrow \text{Interpolate}\left(t, A, B, s = m^2_{i,j}, (\mathsf{sh}_\ell^{i,j,2})_{\ell=j+k-i+2}^{n}\right).$$
7. The share sh_ℓ of P_ℓ is formed by w_ℓ, $\mathsf{sh}_\ell^{i,j,1}$ for $i \in [k]$ and $\max\{i+1, \ell+1\} \leq j \leq \min\{n-k+i, \ell+i\}$, and $\mathsf{sh}_\ell^{i,j,2}$ for $i \in [k]$ and $\max\{i+1, \ell-k+i-1\} \leq j \leq \min\{n-k+i, \ell-1\}$.

[a] If $j = i+1$ then $A = \emptyset$ and Procedure Interpolate gives the secret (i.e., $m^1_{i,j}$) to the first $j-1$ parties; this is a 1-out-of-$(j-1)$ scheme as required.

Fig. 7. A CSSS realizing a partially defined $(n-k)$-slice f.

Proof. By Remark 2.5, it suffices to realize partially-defined slice functions, only defined on inputs of weight $n-k$. The CSSS for such a function f is described in Fig. 7.

We first elaborate on the assumptions used in the scheme and prove that the running time of the sharing algorithm is polynomial in the size of the representation of the slice function and the security parameter (as required in Definition 2.8). In the scheme, we assume that the PRG is $\text{poly}(t(\lambda))$-secure. By [HILL99], such PRG can be built from a $t(\lambda)$-secure one-way function and its running time for a pseudorandom string of length knc is $knc \cdot \text{poly}(\lambda)$. Furthermore, if we use the CCDS protocol of [ABI$^+$23b], the message size is $c = O(\lambda \log n)$ and the running time of the encoding algorithm (for computing the nk messages) is $n^k \cdot \text{poly}(\lambda)$ (see Theorem 2.10). Finally, interpolation can be implemented using $\tilde{O}(n)$ arithmetic operations over a field $\mathbb{F}_p$ (using FFT), where $\log p \approx c = O(\lambda \log n)$; each arithmetic operation can be performed in time $\tilde{O}(\log p) = \tilde{O}(\lambda \log n)$ (using Schönhage–Strassen multiplication algorithm). As the PRG is executed n times and there are $O(nk)$ interpolations, the total running time of the sharing is $n^k \cdot \text{poly}(\lambda) + \tilde{O}\ (n^2k) \cdot \text{poly}(\lambda)$. As the size of the representation of the $(n-k)$-slice function f is $\binom{n}{k(n)} \geq (n/k)^k \geq \sqrt{n}^k = (n^k)^{0.5}$ (as $k(n) \leq \sqrt{n}$), the running time is polynomial in the representation and the security parameter.

We next prove the correctness and security of the scheme, only considering inputs of weight $n-k$. This scheme is an optimization of the perfect secret-sharing scheme described in Fig. 4, where we use a CCDS protocol instead of a perfect CDS protocol and use procedure Interpolate to share the shares, using correlated pseudorandom strings as the "random shares"; however these are valid shares in Shamir's secret-sharing scheme. Thus, the correctness follows from the correctness of the scheme from Fig. 4.

For the security of the scheme described in Fig. 7, we assume that there is an adversary $\mathcal{A}_{\text{CSSS}}$ trying to break the CSSS and prove that if $\mathcal{A}_{\text{CSSS}}$ succeeds then there is either an adversary breaking the CCDS protocol or an adversary breaking the PRG, contradicting their security. Recall that $\mathcal{A}_{\text{CSSS}}$, on input 1^λ, chooses an $(n-k)$-slice function f and an input $x \in \{0,1\}^n$ such that $\text{wt}(x) = n-k$ and $f(x) = 0$ and gets from the challenger shares $(\mathsf{sh}_i)_{x_i=1}$ generated by the scheme for a random secret $s \in \{0,1\}$.

We define $n+1$ hybrids, where in the d-th hybrid a secret s is chosen with uniform distribution, and the shares given to the adversary are generated similar to the scheme in Fig. 7, where we replace some pseudorandom strings in step 1 with truly random strings as follows: For every $1 \leq \ell \leq d$, if $x_\ell = 0$ we use for every $i \in [k], j \in [n]$ a truly random string for $y_\ell^{i,j}$ (instead of $y_\ell^{i,j} \leftarrow \text{PRG}_{i,j}(w_\ell)$). All other strings $y_\ell^{i,j}$ are generated as pseudorandom strings. Let pr_d be the probability that the adversary $\mathcal{A}_{\text{CSSS}}$ guesses the secret given the shares generated in the d-th hybrid for a uniformly distributed secret $s \in \{0,1\}$.

Notice that in the 0-th hybrid the shares given to the adversary are generated as in the scheme described in Fig. 7. On the other hand, in the n-th hybrid, for all parties not in I_x, the shares are generated using truly random strings. In this case, we will show that by the security of the CCDS protocol, the probability that the adversary guesses s, i.e., pr_n is at most $1/2 + 1/\operatorname{poly}(t(\lambda))$.

Claim 4.2. *Assume that the CCDS protocol used in the protocol described in Fig. 4 is $t^{\alpha}(\lambda)$-secure for some constant $\alpha < 1$ and that $\mathcal{A}_{CSSS}$ runs in time $t^{0.4\alpha}(\lambda)$. Then, $\mathrm{pr}_n \leq 1/2 + 1/t^{\alpha}(\lambda)$.*

Proof. Given the adversary $\mathcal{A}_{\mathrm{CSSS}}$ we construct the following adversary $\mathcal{A}_{\mathrm{CCDS}}$ against the CCDS protocol:

Input: 1^{λ}.

- Invoke $\mathcal{A}_{\mathrm{CSSS}}$ with 1^{λ} and get an $(n-k)$-slice function $f : \{0,1\}^n \to \{0,1\}$ for some n and an input $x \in \{0,1\}^n$ such that $\mathrm{wt}(x) = n-k$ and $f(x) = 0$; the function f is described by a truth table of size $\binom{n}{k}$.
- Construct g_f from f and compute the indices $j_1, \ldots, j_k$, where $x_{j_1} = \cdots = x_{j_k} = 0$ and $j_1 < \cdots < j_k$ and send g_f and $j_1, \ldots, j_k$ to the CDS Challenger.
- Get messages $(m_{i,j_i})_{i \in [k]}$ from the challenger, where m_{i,j_i} is the message of the i-th server on input j_i and a randomly chosen secret $s \in_U \{0,1\}$ (the same secret is used to generate all messages).
- For every $i \in [k]$ and $j \neq j_i$ choose $m_{i,j} = 0^c$.
- Compute shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ using steps 1, 3 – 7 of the scheme described in Fig. 7.
- Send $(\mathsf{sh}_i)_{x_i=1}$ to $\mathcal{A}_{\mathrm{CSSS}}$, get a secret s', and output it.

Assume that $\mathcal{A}_{\mathrm{CSSS}}$ outputs the correct secret in the n-th hybrid with probability $1/2 + \varepsilon(\lambda)$. First, we argue that the probability that $\mathcal{A}_{\mathrm{CCDS}}$ outputs the correct secret, i.e., $s = s'$, is $1/2 + \varepsilon(\lambda)$. For every $i \in [k]$ the message m_{i,j_i} is generated as in the scheme described in Fig. 7 with the secret s chosen by the CCDS challenger. Fix $i \in [k]$ and $j \neq j_i$ and consider the shares given to I_x of $m_{i,j}$ as generated steps 3 – 7 of the scheme described in Fig. 7. As proved in Lemma 3.4, in these shares either for $m^1_{i,j}$ or for $m^2_{i,j}$ the set I_x has less than the appropriate threshold of shares. Thus, even if some shares of the threshold secret-sharing scheme that are given to I_x are generated using a pseudorandom string, the shares given to I_x of 0^c and the "real message" $m_{i,j} = \mathrm{CCDS.E{\scriptstyle NC}}(1^{\lambda}, g_f, i, j; r)$ are equally distributed. Thus, the shares given to $\mathcal{A}_{\mathrm{CSSS}}$ when it attacks the CSSS described in Fig. 7 and the shares $\mathcal{A}_{\mathrm{CSSS}}$ gets from $\mathcal{A}_{\mathrm{CCDS}}$ are equally distributed and the probability that $\mathcal{A}_{\mathrm{CCDS}}$ outputs $s' = s$ is the probability that $\mathcal{A}_{\mathrm{CSSS}}$ outputs $s' = s$, i.e., $\mathrm{pr}_n = 1/2 + \varepsilon(\lambda)$.

The running time of $\mathcal{A}_{\mathrm{CCDS}}$ is the running time of $\mathcal{A}_{\mathrm{CSSS}}$, the time it takes to translate f to g_f, and the running time of steps 3 – 7 in the scheme described in Fig. 7. The running time of $\mathcal{A}_{\mathrm{CSSS}}$ is $t^{0.4\alpha}(\lambda)$. The size of the description of $g_f : [n]^k \to \{0,1\}$ is n^k. On the other hand, the size of the description of f, i.e., the size of its truth table on inputs of weight $n-k$, is $\binom{n}{k} \geq (n/k)^k \geq \sqrt{n}^k = (n^k)^{0.5}$ (where we use the fact that $k \leq \sqrt{n}$). Since the adversary $\mathcal{A}_{\mathrm{CSSS}}$ runs in time $t^{0.4\alpha}(\lambda)$, we deduce that $\binom{n}{k} \leq t^{0.4\alpha}(\lambda)$. Thus, the time required to write

g_f is at most $t^{0.8}(\lambda)$. Furthermore, as analyzed above, the running time of steps 3 – 7 is $\tilde{O}(n^2k) \cdot \text{poly}(\lambda)$. Since $k \geq 2$,

$$\tilde{O}(n^2k) \leq \tilde{O}(n^3) \leq n^{2k}.$$

Furthermore, since $1/t(\lambda)$ is negligible, $\text{poly}(\lambda) \leq t^{0.1\alpha}(\lambda)$. Thus, the running time of steps 3 – 7 is $\tilde{O}(n^2k) \cdot \text{poly}(\lambda) \leq n^{2k}t^{0.1\alpha}(\lambda) \leq t^{0.9\alpha}(\lambda)$. We conclude that the running time of $\mathcal{A}_{\text{CCDS}}$ is less than $t^{\alpha}(\lambda)$. By the $t^{\alpha}(\lambda)$-security of the CCDS protocol, the probability that $\mathcal{A}_{\text{CCDS}}$ outputs $s = s'$ is at most $1/2 + 1/t^{\alpha}(\lambda)$; by the discussion above this is also true for $\mathcal{A}_{\text{CSSS}}$. ■ (of Claim 4.2)

We next show that, by the security of the PRG, the probability that an adversary guesses the secret in hybrid d is at most $1/t^{\alpha}(\lambda)$ greater than the probability that it guesses the secret in hybrid $d + 1$.

Claim 4.3. *Assume that the PRG used in the protocol described in Fig. 4 is $t^{\alpha}(\lambda)$-secure for some constant $\alpha < 1$ and that $\mathcal{A}_{CSSS}$ runs in time $t^{0.4\alpha}(\lambda)$. Then, $\text{pr}_{d-1} - \text{pr}_d \leq 1/t^{\alpha}(\lambda)$ for every $1 \leq d \leq n$.*

Proof. The two hybrids differ only in the way that $y_d^{i,j}$ is generated. Thus, if $x_d = 1$, the hybrids are the same and the claim follows. In the following we assume that $x_d = 0$. Given the adversary $\mathcal{A}_{\text{CSSS}}$ we construct the following adversary $\mathcal{A}_{\text{PRG}}$ against the PRG:

Input: 1^{λ} and a random or pseudorandom string $y = (y^{i,j})_{i \in [k], j \in [n]}$, where $y^{i,j} \in \{0,1\}^c$.

- Invoke $\mathcal{A}_{\text{CSSS}}$ with 1^{λ} and get an $(n-k)$-slice function $f : \{0,1\}^n \to \{0,1\}$ for some n and an input $x \in \{0,1\}^n$ such that $\text{wt}(x) = n - k$ and $f(x) = 0$; the function f is described by a truth table of size $\binom{n}{k}$.
- Choose a uniformly distributed secret $s \in \{0,1\}$.
- For every $\ell \in [n]$, sample with uniform distribution a seed $w_\ell \in \{0,1\}^{\lambda}$.
- For every $\ell \in [n]$ and every every $i \in [k], j \in [n]$,
  - If $x_\ell = 0$ and $\ell \leq d - 1$, then choose $y_\ell^{i,j}$ with uniform distribution from $\{0,1\}^c$.
  - If $\ell = d$, then $y_\ell^{i,j} = y^{i,j}$.
  - If $x_\ell = 1$ or $\ell \geq d + 1$, then $y_\ell^{i,j} \leftarrow \text{PRG}_{i,j}(w_\ell)$.
- Compute shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ for the secret s using steps 2 – 7 of the scheme described in Fig. 7.
- Send $(\mathsf{sh}_i)_{x_i=1}$ to $\mathcal{A}_{\text{CSSS}}$, get a secret s'.
- If $s = s'$ output 1 else output 0.

As in the proof of Claim 4.2, the running time of $\mathcal{A}_{\text{PRG}}$ is less than $t^{\alpha}(\lambda)$, thus, by the $1/t^{\alpha}(\lambda)$-security of the PRG, the probabilities that $\mathcal{A}_{\text{PRG}}$ outputs 1 when y is pseudorandom and when y random can differ by at most $1/t^{\alpha}(\lambda)$. Note that if y is pseudorandom then the shares are generated as in hybrid $d-1$ and the probability that $s = s'$ is the probability that $\mathcal{A}_{\text{CSSS}}$ outputs the correct secret from the shares generated in hybrid $d - 1$, i.e., pr_{d-1}. Similarly, if y is random then the shares are generated as in hybrid d and the probability that $s = s'$ is the probability that $\mathcal{A}_{\text{CSSS}}$ outputs the correct secret from the shares generated in hybrid d, i.e., pr_d. Combining the above 3 facts, we deduce that $\text{pr}_{d-1} \leq \text{pr}_d + 1/t^{\alpha}(\lambda)$ as claimed. ■ (of Claim 4.3)

We have assumed that the PRG and the CCDS protocol used in the scheme of Fig. 7 are $1/t^{\alpha}(\lambda)$ secure for some constant $\alpha < 1$. Consider an adversary $\mathcal{A}_{\text{CSSS}}$ that runs in time $t^{0.4\alpha}(\lambda)$. By Claim 4.2 and Claim 4.3,

$$\text{pr}_0 = (\text{pr}_0 - \text{pr}_1) + \cdots + (\text{pr}_{n-1} - \text{pr}_n) + \text{pr}_n \leq \frac{n}{t^{\alpha}(\lambda)} + \frac{1}{2} + \frac{1}{t^{\alpha}(\lambda)}.$$

Thus, pr_0 – the probability that an adversary guesses the secret in the 0-th hybrid i.e., in the scheme described in Fig. 7 – is at most $1/2 + \frac{n+1}{t^{\alpha}(\lambda)}$. As $n \leq \lambda$ and $1/t(\lambda)$ is a negligible function , this probability is less than $1/2 + 1/t^{0.4\alpha}(\lambda)$. To conclude, we have proved that the probability that any adversary running in time at most $t^{0.4\alpha}(\lambda)$ guesses the secret with probability at most $1/2+1/t^{0.4\alpha}(\lambda)$, i.e., the CSSS is $1/t^{0.4\alpha}(\lambda)$-secure.

We complete the proof by analyzing the share size. There are $O(kn)$ executions of threshold t'-out-of-n' secret-sharing schemes in the scheme described in Fig. 7; in each one of them $n' - t' \leq k - 1$. In each such scheme there are at most k "extra" shares. The scheme distributes these "extra" shares such that each party gets $O(k^2)$ "extra" shares, i.e., in the scheme for $m^1_{i,j}$, which is shared among the first $j - 1$ parties only the last $i - 1$ parties $P_{j-i}, \ldots, P_{j-1}$ get the "extra" shares and in the scheme for $m^2_{i,j}$, which is shared among the last $n-j-$ parties only the first $k - i$ parties $P_{j+1}, \ldots, P_{j+k-i+1}$ get the "extra" shares. To conclude, the share of each party contains one seed of size λ and $O(k^2)$ shares in Shamir's scheme with secrets of length $O(\lambda \log n)$ – the message size of the CCDS protocol of [ABI+23b] (see Theorem 2.10). All together, the share size of each party is $O(k^2 \lambda \log n)$. □

5 Applications to Multislices

In this section, we present implications for multislices of the improved schemes for $(n - k)$-slices presented in Sections 3 and 4. We split the results into different subsections. In Section 5.1, we present some results we use about the construction of secret-sharing schemes from monotone Boolean formulas. In Section 5.2, we present general results for $(a : b)$-multislices, proving Theorem 1.5. For $(n : n-k)$-multislices with $k = o(n)$, we are able to find better schemes, and the rest of the section is dedicated to this case. In Section 5.3, we present better upper bounds on the share size for perfect schemes and we prove Theorem 1.6. Section 5.4 is dedicated to CSSSs and Section 5.5 is dedicated to linear and multi-linear secret-sharing schemes.

5.1 The Framework

First, we discuss the framework in which we use the schemes for $(n-k)$-slices. Many general and multislice schemes from recent years are based on the following paradigm:

1. Given a function f over n bits, build a constant-depth formula F for f that uses AND and OR gates, together with gates that compute (k,N)-slice functions for some N, and $k=O(\log N)$. In these formulas, all (k,N)-slice gates are in the same level, that is, in every path from the root to a leaf there is at most one (k,N)-slice gate.
2. Apply the closure properties of secret-sharing schemes over formulas to realize a scheme for f according to F (Lemma 5.2). Whenever necessary, plug in a black-box way an efficient scheme for k-slices (or k-server CDS protocols) that has shares of size $2^{\tilde{O}(\sqrt{k \log N})}$[LVW18].

Given a scheme for a multislice function f that follows the above paradigm, we will be able to use the following simple duality lemmas to realize the dual of f. The first lemma is a folklore result that was stated and proved in [ABN$^+$22]. The second one is a natural generalization of the [BL88] whose proof also appears in [ABN$^+$22].

Lemma 5.1 (Formulas and Duality). *Let C be a formula that computes a function $f:\{0,1\}^n \rightarrow \{0,1\}$ and let $G_1,\ldots,G_k$ be its gates. For any gate G that computes a function g, denote by G^* a gate that computes g^*. Then, a formula C' with the same structure as C and with every gate G_i replaced with G_i^* computes the dual function f^*.*

Lemma 5.2 (Formulas and Secret Sharing). *Suppose that a monotone function $f:\{0,1\}^n \rightarrow \{0,1\}$ can be implemented by a formula F over some collection of monotone gates G, and assume that every gate $g \in G$ can be realized by a secret-sharing scheme whose share-size is w_g. Then, f can be realized by a secret-sharing scheme whose share size is $\max_{i \leq i \leq n}\{w_{F,i}\}$, where the weight function $w_{F,i}$ is defined as follows.*

- *The weight $w_F(v)$ of a leaf v in F is the product $\prod_j w_{g_j}$ where g_j is the jth gate in the (unique) path from the root to v.*
- *The weight $w_{F,i}$ of the ith variable in the formula F is the sum of $w_F(v)$ of all leaves v labeled by x_i.*

Similarly, if every gate g can be realized by a secret-sharing scheme whose information ratio is w'_g, then f can be realized by a secret-sharing scheme with information ratio of $\max_{i \leq i \leq n}\left\{w'_{F,i}\right\}$.

It is worth noticing that if we consider a formula F whose gates can be realized by a secret-sharing schemes with good information ratio, then the resulting share size of party P_i will be between $w_{F,i}$ and $w'_{F,i}$. For instance, if F has k layers of t-out-of-n threshold gates, the share size increases by $\log n$ in each path, but not by $\log^k n$.

Remark 5.3. Previous constructions of secret-sharing schemes, starting from the work of Liu and Vaikuntanathan [LV18], use CDS protocols. When viewed as a formula, this is translated to a so-called CDS gate. We will not define these gates in this paper; we rather use the observation from [ABN$^+$22] that each such gate is a slice function, specifically a k-server CDS protocol for a function $f : [N]^k \to \{0,1\}$ is translated to k-slice function with kN variables. In the construction of linear and multi-linear secret-sharing schemes we will need the fact that if there is a CDS protocol for f with message size $c_{\text{cds}}(k,N)$, then the k-server CDS gate can be realized by a secret-sharing scheme in which the share size of each party is $c_{\text{cds}}(k,N)$.

When proving our results for $(n-k : n)$-multislices, we will also use the following well-known result on the duality of linear and multi-linear schemes:

Theorem 5.4 ([Gál95,Feh98,FHKP17]). *If f can be realized by a linear secret-sharing scheme over $\mathbb{F}_q$ with share size $r \log q$, then f^* can be realized by a linear secret-sharing scheme over $\mathbb{F}_q$ with share size $r \log q$.*

If f can be realized by a multi-linear secret-sharing scheme over $\mathbb{F}_q$ with information ratio r, then f^* can be realized by a multi-linear secret-sharing scheme over $\mathbb{F}_q$ with information ratio r (and the same domain of secrets).

5.2 Schemes for $(a : b)$-Multislices

We restate and prove Theorem 1.5, which implies that the current gap between the share sizes of the family of $(a : b)$-multislices and its dual can be narrowed down to $2^{o(n)}$. Recall that this has implications on the share size of random hypergraph access structures (Corollary 1.7).

Theorem 5.5 (Share Size of Multislices, Theorem 1.5 Restated). *For every $a < b \in [n]$, every $(a : b)$-multislice can be realized by a secret-sharing scheme with share size $\binom{n-a}{\geq n-b} \cdot 2^{o(n)}$.*

Proof. As analyzed by [ABN$^+$22], the scheme of Applebaum and Nir [AN21] for $(a : b)$-multislices that has share sizes $\binom{b}{\geq a} \cdot 2^{o(n)}$ works according to the paradigm specified in the introduction of this section. For each multislice, they construct a formula F over AND and OR gates, combined with gates that compute (k,N)-slice gates with $k = \sqrt{n}$, $N = \sqrt{n}2^{\sqrt{n}}$.

By Lemma 5.1, if we replace every gate G in F by its dual gate G^* we get a formula F^* that computes f^*, the dual of f. This formula will consist of AND and OR gates, together with $(N-k,N)$-slice gates with $k = \sqrt{n}$, $N = \sqrt{n}2^{\sqrt{n}}$, which are the duals of the slice gates that appeared in F. By Lemma 5.2, the overhead of realizing a secret-sharing scheme for f^* based on F^* compared to realizing f based on F boils down solely to the difference in the cost of the different types of slice gates used in each of the formulas. In [ABNP20] the (k,N)-slices of F with the above parameters were realized with share size $2^{\tilde{O}(\sqrt{n})}$; by Theorem 3.5 we get the same asymptotical share size for the $(N-k,N)$-slices of F^*, i.e., we get share size $kN2^{\tilde{O}(\sqrt{k \log N})} = 2^{\tilde{O}(\sqrt{n})}$. Hence, since the scheme of [ABNP20] realizes $(n-b:n-a)$-multislices with shares of size $\binom{n-a}{\geq n-b}\cdot 2^{o(n)}$, their dual $(a:b)$-multislices can also be realized with the same share size, as desired. □

5.3 Schemes for $(n-k:n)$-Multislices

In the rest of this section, we construct schemes for $(n-k:n)$-multislices that are more efficient than the ones described in Theorem 1.5 when $k=o(n)$. Our main result for perfect schemes is in Theorem 5.18. The current best upper and lower bounds for $(n-k:n)$-multislices are summarized in Fig. 8.

The share size of perfect schemes for $(n-k:n)$-multislices				
	$k \geq t_1$	$t_1 \geq k \geq t_2$	$t_2 \geq k \geq t_3$	$t_3 \geq k$
Upper bounds	$2^{0.59n}$ [AN21]	$O(\binom{n-1}{k-1})$	$2^{O(k)}$ Theorem 5.12	$k^{5k}n2^{\tilde{O}(\sqrt{k\log n})}$ Theorem 5.12
Lower bounds	$\Omega(k\log k)$ [Csi97] $\Omega(\log n)$ [BGK16]			

Fig. 8. The best-known bounds on the share size of perfect secret-sharing schemes for $(n-k:n)$-multislices. See Remark 5.23 for more details. The thresholds for the bounds in the table are are $t_1=0.14n$, $t_2=n/\log^2 n$, and $t_3=\log n(\log\log n)^2$.

The schemes for $(n-k:n)$-multislices use our schemes for high slices, constructions for k-hypergraphs from [BF20a] (which are based on robust CDS protocols from [ABNP20]) and an adaptation of constructions for k-hypergraphs from [ABNP20]. In Lemma 5.11, we show how to realize $(n-k:n)$-multislices from secret-sharing schemes for $(n-k)$-slices. The realization has a few stages. In Definition 5.6 we recall a definition from [BF20a] of a special family of functions, called (h_i,k)-hypergraphs. In Lemma 5.7, we show that in order to realize $(n-k:n)$-multislices it suffices to realize the duals of (h_i,k)-hypergraphs. In Lemma 5.9, we quote a result of [BF20a] and in Lemma 5.10, we describe an adaptation of a result of [ABNP20]; these lemmas are used in the proof of Lemma 5.11 to realize the duals of (h_i,k)-hypergraphs from $(n-k)$-slices, i.e., to realize $(n-k:n)$-multislices from $(n-k)$-slices. Lemma 5.10 is proved in Appendix A.

We start by giving some additional notation and presenting some previous results.

Definition 5.6 ((h_i,k)-Hypergraphs). *Let $i\leq k\leq n$ and let $\mathrm{TR}_{k+1,n}$ be the threshold $k+1$ function on n variables. Given an i-hypergraph h_i (i.e., a function whose minterms are of size exactly i), the (h_i,k)-hypergraph is defined as $h_i\vee\mathrm{TR}_{k+1,n}$. That is, in the (h_i,k)-hypergraph all inputs of weight less than i are zeros of f, all inputs of weight greater than k are ones of f, and an input y of weight between i and k is a one of f if and only if there is a minterm x (i.e., an input of weight i for which $f(x)=1$) such that $x\leq y$.*

Note that a (h_i,k)-hypergraph is a (i,k)-multislice, a (h_k,k)-hypergraph is a k-slice, and a (h_i,n)-hypergraph is a hypergraph.

We next show how to realize $(n-k:n)$-multislices from secret-sharing schemes for the dual of (h_i,k)-hypergraphs

Lemma 5.7 (Multislices from duals of (h_i,k)-hypergraphs). *Let f be an $(n-k:n)$-multislice. Assume that for every $i \leq k < n$ and every i-hypergraph h_i, the dual of the (h_i,k)-hypergraph can be realized by a secret-sharing scheme with share size $c_{\text{dual}}(i,k,n)$. Then f can be realized by a secret-sharing schemes with share size $\sum_{i=1}^{k} c_{\text{dual}}(i,k,n)$. If the secret-sharing schemes for the duals of (h_i,k)-hypergraphs are linear, the resulting scheme is linear.*

Proof. As $f(x)=0$ for every x of weight less than $n-k$, the maxterms of f (i.e., the maximal inputs x such that $f(x)=0$) are of weight at least $n-k-1$, thus, f can be written as $f=(\wedge_{i=1}^{k} f_i) \wedge \mathrm{TR}_{n-k,n}$, where f_i is a function whose maxterms are of weight exactly $n-i$. For this proof, it is convenient to describe the access structure as $f=\wedge_{i=1}^{k}(f_i \wedge \mathrm{TR}_{n-k,n})$.

Fix i such that $1 \leq i \leq k$. By Lemma 5.1, the dual of $f_i \wedge \mathrm{TR}_{n-k,n}$ is $f_i^* \vee \mathrm{TR}_{n-k,n}^*$. By definition, f_i^* is a function whose minterms are of weight i, i.e., an i-hypergraph and $\mathrm{TR}_{n-k,n}^* = \mathrm{TR}_{k+1,n}$. We obtain that $f_i^* \vee \mathrm{TR}_{k+1,n}$ is a (f_i^*,k)-hypergraph and $f_i \wedge \mathrm{TR}_{n-k,n}$ is the dual of a (f_i^*,k)-hypergraph. By the assumption of the lemma, $f_i \wedge \mathrm{TR}_{n-k,n}$ can be realized with a shares of size $c_{\text{dual}}(i,k,n)$, thus, f can be realized with shares of size $\sum_{i=1}^{k} c_{\text{dual}}(i,k,n)$ by Lemma 5.2. □

In our schemes, we use a construction of [BF20b] for (h_i,k) hypergraphs (which follows from results of [ABNP20]) and an adaptation of a construction of [ABNP20]. These constructions implicitly follow the framework of Section 5.1, that is, they implicitly construct a formula whose properties are described in Lemma 5.9. For these lemmas we define k-partite slice functions.

Definition 5.8 (k-partite slice functions). *A (k,kN)-slice $g:[kN] \to \{0,1\}$ is a (k,kN)*-partite slice *if for every input x of weight k such that $g(x)=1$ and for every $j \in [k]$*

$$|\{\ell : x_\ell = 1\} \cap \{(j-1)N+1,\ldots,jN\}| = 1,$$

that is, the variables are partitioned to k subsets of size N and in each minterm of weight k there is exactly one satisfied variable in each subset.

Notice that (k,kN)-partite slices are called CDS gates in [ABN$^+$22]; they are basically equivalent to k-server CDS protocols (defined in [GIKM00]) and can be realized by a secret-sharing scheme with share size $2^{O(\sqrt{k\log N}\log\log N)}$ [LVW18].

Lemma 5.9 ([BF20a, Lemma 7.2 (implicit)]). *Let h_i be an i-hypergraph with n variables, and let $i \leq k \leq \min\{i \cdot n/2, 2^{\sqrt{n/i}}\}$ be an integer. Then, there is a constant depth monotone formula for the (h_i,k)-hypergraph with AND, OR, threshold, and (i,ni)-partite slices, whose size is $\ell_i = O(i^{3i}2^i k^i \log^{2i-1} k \log^2 n)$. Furthermore, all the (i,ni)-partite slice gates are in the same level.*

Lemma 5.10. *Let h_i be an i-hypergraph with n variables, and let k be an integer such that* $\max\{i, \log n \cdot \log\log n\} \leq k \leq \min\{i \cdot n/2, 2^{\sqrt{n/i}}\}$*. Then, there is a constant depth monotone formula for the (h_i, k)-hypergraphs with AND, OR, threshold, and $(O(k/\log k), n^{O(\log k)})$-partite slice gates, whose size is $\ell_i = 2^{O(k)}$. Furthermore, all the $(O(k/\log k), n^{O(\log k)})$-partite slices are in the same level.*

The proof of Lemma 5.10 can be found in Appendix A.

Lemma 5.11 (Multislices via a formula with partite-slice gates). *Let $k < n$. Assume that for every $i \leq k$, for every (h_i, k) hypergraph there is a formula of size ℓ_i with AND, OR, threshold, and $(\kappa_i, \kappa_i N_i)$-partite slice gates and all slices are in the same level. Furthermore, assume that for every $1 \leq i \leq k$ every $(\kappa_k N_k - \kappa_k, \kappa_k N_k)$-slice can be realized by a secret-sharing scheme with 1-bit secrets and share size $c_{\text{slice}}(\kappa_i, \kappa_i N_i, 1) \leq c_{\text{slice}}(\kappa_k, \kappa_k N_K, 1)$. Then every $(n - k : n)$-multislice f can be realized by a secret-sharing scheme with 1-bit secrets and share size*

$$\sum_{i=1}^{k} O(c_{\text{slice}}(\kappa_k N_k - \kappa_k, \kappa_k N_k, 1)\ell_i \operatorname{polylog}(n)).$$

If the secret-sharing schemes for the $(N_i\kappa_i - i, \kappa_i N_i)$-slices are linear, the resulting scheme is linear.

Proof. By Lemma 5.7, in order to construct a secret-sharing scheme for f it suffices to show how to realize duals of (h_i, k) hypergraphs.

We realize the dual of an (h_i, k) hypergraph by the formula-based secret-sharing scheme of Lemma 5.2 using the dual of the formula that is assumed in the lemma. By Lemma 5.1 the dual of the formula computes the dual of the (h_i, k) hypergraph; this formula has size ℓ_i, constant depth, and its gates are AND, OR, threshold, and dual of $(\kappa_i, \kappa_i N_i)$-partite slices. As every $(\kappa_i, \kappa_i N_i)$-partite slice is, in particular, a $(\kappa_i, \kappa_i N_i)$-slice, its dual is a $(\kappa_i N_i - \kappa_i, \kappa_i N_i)$-slice.

Recall that for a one-bit secret, AND and OR functions can be realized by a secret-sharing scheme with share size 1, threshold functions can be realized by a scheme with share size $\log n$, and $(\kappa_i N_i - \kappa_i, \kappa_i N_i)$-slices can be realized by a scheme with share size $c_{\text{slice}}(\kappa_i N_i - \kappa_i, \kappa_i N_i, 1)$. As all slice functions are in the same level, by Lemma 5.2, the dual of every (h_i, k) hypergraph can be realized by a secret-sharing scheme with one-bit secrets and share size $O(\ell_i c_{\text{slice}}(\kappa_i N_i - \kappa_i, \kappa_i N_i, 1) \operatorname{polylog}(n))$.

By Lemma 5.7, every $(n - k : n)$ multislice can be realized by a secret-sharing scheme with one-bit secrets and share size

$$\sum_{i=1}^{k} O(c_{\text{slice}}(\kappa_i N_i - \kappa_i, \kappa_i N_i, 1)\ell_i \operatorname{polylog}(n))$$
$$\leq \sum_{i=1}^{k} O(c_{\text{slice}}((\kappa_k N_k - \kappa_k, \kappa_k N_k)\ell_i \operatorname{polylog}(n)).$$

□

Theorem 5.12 (Share Size of $(n-k:n)$-Multislices, Theorem 1.6 Restated). *For $k \leq \log n(\log\log n)^2$, every $(n-k:n)$-multislice can be realized by a secret-sharing scheme with share size $k^{5k} \cdot n \cdot 2^{\tilde{O}(\sqrt{k \log n})}$. For $\log n(\log\log n)^2 < k < n/\log^2 n$, every $(n-k:n)$-multislice can be realized by a secret-sharing scheme with share size $2^{O(k)}$.*

Proof. The theorem is a consequence of Lemma 5.11, taking schemes for slice functions from Theorem 3.5.

For $k < \log n(\log\log n)^2$, we use formula of Lemma 5.9 with $\kappa_i = i$, $N_i = n$, and $\ell_i = O(i^{3i} 2^i k^i \log^{2i-1} k \log^2 n)$. Observe that $\sum_{i=1}^{k} \ell_i = O(\ell_k)$. We next calculate the resulting share size in this case.

$$\begin{aligned}
&\sum_{i=1}^{k} O(c_{\text{slice}}(nk-k, nk, 1) \cdot \ell_i \operatorname{polylog}(n)) \\
&\qquad \leq \tilde{O}(c_{\text{slice}}(nk-k, nk, 1) \cdot \ell_k) \\
&\qquad = k^2 n 2^{\tilde{O}(\sqrt{k \log n})} \cdot O(k^{3k} 2^k k^k \log^{2k-1} k \log^2 n) \\
&\qquad = \tilde{O}(n) \cdot 2^{\tilde{O}(\sqrt{k \log n})} \cdot \left(k^{5k} \cdot \frac{2^k \log^{2k-1} k^3}{k^k}\right) \\
&\qquad = k^{5k} n 2^{\tilde{O}(\sqrt{k \log n})}.
\end{aligned}$$

For $\log n(\log\log n)^2 < k < n/\log^2 n$, we use formula of Lemma 5.10 with $\kappa_i = O(k/\log k)$, $N_i = n^{O(\log k)}$, and $\ell_i = 2^{O(k)}$. Observe that $\sum_{i=1}^{k} \ell_i = 2^{O(k)}$. Furthermore, by Theorem 3.5,

$$\begin{aligned}
&c_{\text{slice}}(n^{O(\log k)} - O(k/\log k), n^{O(\log k)}, 1) \\
&\qquad = n^{O(\log k)} \cdot (k/\log k) \cdot 2^{O(\sqrt{(k/\log k)\log(n^{O(\log k)})}\log(k/\log k \cdot \log(n^{O(\log k)})))} \\
&\qquad = 2^{O(\log k \log n + \sqrt{k \log(n)}(\log k + \log\log n))}.
\end{aligned}$$

When $k > \log n(\log\log n)^2$, the above expression is at most $2^{O(k)}$. The resulting share size in this case is as follows.

$$\sum_{i=1}^{k} O(c_{\text{slice}}(n^{O(\log k)} - O(k/\log k), n^{O(\log k)}, 1) \cdot \ell_i \operatorname{polylog}(n)) = 2^{O(k)}.$$

□

Remark 5.13. In Theorem 5.12, we implicitly construct a constant depth formula for $(n-k:n)$-multislices of size $O(k\ell_k)$ with AND, OR, threshold, and dual of partite slice gates, where the dual of partite slice gates are in the same level. That is, we construct a formula for duals of (h_i, k)-hypergraphs (using the formulas of Lemma 5.9 or Lemma 5.2)and construct from it a formula for $(n-k:n)$-multislices. In most of our constructions, we replace the dual of $(\kappa_i N_i - \kappa_i, \kappa_i N_i)$-partite slice gates by $(\kappa_i, \kappa_i N_i)$-slices.

Remark 5.14. In Theorem 5.12, we can improve the share size to $2^{\tilde{O}(\sqrt{k \log n})}$ when $\log n \log \log n < k \leq \log n (\log \log n)^2$ by using the formula of Lemma 5.10. To simplify the theorem, we ignore this case.

5.4 CSSS for $(n-k:n)$-Multislices

We next use the construction of Section 5.3 to construct CSSS for $(n-k:n)$-multislices. For the CSSS we use a computational analogue of the formula-based perfect secret-sharing scheme of Benaloh and Leichter [BL88]; in our scheme we use CSSS to realize the gates. Proving the security of the resulting CSSS requires analyzing the representation size of the gates and the share size as well as running time of the CSSS implementing the gates. We prove the security for the specific formula of Lemma 5.2. We remark that in the computational setting, Yao [Yao89] showed that a CSSS can use a monotone *circuit*; however, we do not need this generalization.

Remark 5.15. The CSSS for $(n-k:n)$-multislices as described in Theorem 5.16 is interesting when $k \leq \log n$. For $\log n < k < \log n(\log\log(n))^2$, it has share size that is worse than the perfect secret-sharing scheme described in Theorem 5.12 by a factor of λ. Furthermore, for $k > \log n(\log\log(n))$, we can use the formula of Lemma 5.10 and obtain a CSSS with share size $O(2^{O(k)}\lambda \operatorname{polylog}(n))$. As this secret-sharing scheme will only outperform the perfect secret-sharing scheme described in Theorem 5.12 in a small range of parameters, we omit these details.

Theorem 5.16 (CSSSs for $(n-k:n)$-Multislices). *Let $k \leq \sqrt{n}$ and $t(\lambda) \geq k^{ck} \cdot \operatorname{polylog}(n)$ for a sufficiently large constant c. Assuming the existence of $t(\lambda)$-secure OWFs, there is a $\operatorname{poly}(t(\lambda))$-secure CSSS for $(n-k:n)$-multislices, represented as a truth table of all inputs of weight at least $n-k$. The share size in the CSSS is $O(k^{5k}\lambda \operatorname{polylog}(n))$, where λ is the security parameter, and the running time of the sharing and reconstruction algorithms of the CSSS is $\tilde{O}(n^{4k}) \cdot \operatorname{poly}(\lambda) = \operatorname{poly}(\binom{n}{k}, \lambda)$.*

Proof. The construction of the CSSS is the same as the construction used to prove Lemma 5.2, i.e., a generalization of the formula-based secret-sharing scheme of [BL88]. To prove the security of the CSSS we will need to explicitly describe it as some of the details of the scheme require care. We use the formula for $(n-k:n)$-multislices as discussed in Remark 5.13. This is a constant depth formula of size $O(k\ell_k)$ with AND, OR, threshold, and $(in-i, in)$-slice gates for some $1 \leq i \leq k$. By Theorem 4.1, assuming the existence of $t(\lambda)$-OWFs, there are $t^{\alpha}(\lambda)$-secure CSSSs for these functions with share size $O(k^2\lambda \log n)$.

We next describe the CSSS. To share a secret $s \in \{0,1\}$ for an $(n-k:n)$-multislice f, we use the following recursive procedure:

- We start from the root of the formula for f and the secret s.
- In each recursive call, we are at some gate G and have some secret s_G generated for the gate.
  - If G is a leaf labeled by some variable x_i, we give s_G to party P_i.

- Otherwise, we share s_G using a CSSS realizing G, producing shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_k$, where k is the arity of the gate.
- For each $1 \leq j \leq k$, the share of the gate G_j, where G_j is the j-th child of G, is sh_j. We apply the procedure recursively for G_j and sh_j.

To complete the description of the scheme, we need to elaborate on the CSSS for each gate. For AND, OR, and threshold gates we use information-theoretic secret-sharing realizing them. For a gate computing an $(in - i, in)$-slice function g we use the CSSS of Theorem 4.1. Note that in this case the secret for the CSSS for the $(kn - k)$-slice is composed of many bits and we share each bit of the secret independently using the CSSS of Theorem 4.1.[17]

The correctness follows by the same arguments as in [BL88] as we next explain. For every gate G in the formula computing f, let f_G be the function computed by the sub-formula rooted at the gate. For every input x such that $f(x) = 1$, by staring from the leaves and going to the root, the parties in I_x can reconstruct the secret of the gate G for every gate G such that $f_G(x) = 1$; in particular they can reconstruct the secret of the root, i.e., the secret shared in the CSSS. The share size of this scheme is computed as in Lemma 5.2, i.e., denoting the share size in the CSSS for slices by $c_{\text{cslice}}(k, n, 1) = \tilde{O}(k^2\lambda)$, the share size is

$$O(c_{\text{cslice}}(k, nk, 1) \cdot k\ell_k \cdot \text{polylog}(n)) = O(k^{5k} \cdot \lambda \cdot \text{polylog}(n)).$$

The running time of the sharing and reconstruction algorithms of the CSSS is $O(\ell_k k \cdot \text{polylog}(n))$ times the running time of the sharing and reconstruction algorithms for the CSSS for $(nk - k, nk)$-slices, i.e., it is $\tilde{O}(k^{5k}(nk)^k) \cdot \text{poly}(\lambda) = \tilde{O}(n^{4k}) \cdot \text{poly}(\lambda) = \text{poly}(\binom{n}{k}, \lambda)$ (since $k \leq \sqrt{n}$).

We next argue that the CSSS we constructed is $\text{poly}(t(\lambda))$-secure assuming the CSSS for $(kn - k)$-slices are $\text{poly}(t(\lambda))$-secure. That is, we assume that there is an adversary $\mathcal{A}_{\text{multislice}}$ breaking the CSSS for $(n - k : n)$-multislices and construct an adversary $\mathcal{A}_{\text{slice}}$ breaking the CSSS for $(n - k : n)$-slices.

Given a security parameter λ, let f be the multislice chosen by $\mathcal{A}_{\text{multislice}}$, let n be the input size of f, and let $x \in \{0, 1\}^n$ such that $f(x) = 0$ be the input chosen by $\mathcal{A}_{\text{multislice}}$. We start with some notation. Consider the formula for f and assume that it has $m = O(\ell_k)$ gates $G_1, \ldots, G_m$, where we order the gates such that if the output of gate G_i is an input of gate G_j then $i < j$, e.g., G_m is the root and G_1 is a leaf labeled by some variable x_j. Finally, for $1 \leq i \leq m$, let f_i be the function computed by the sub-formula rooted at G_i.

We define $m + 1$ hybrids. Recall that the CSSS for f computes secrets for the gates, starting with the root G_m, then a secret for G_{m-1}, and so on until it computes secrets for the leaves. In every non-leaf gate G_i, the CSSS shares the secret of the gate, and the shares are the secrets of the children of the gate. The share of party P_j is the secret of all laves labeled by x_i.

[17] By a simple hybrid argument, this is a CSSS for long secrets; in such a CSSS the adversary sends two secrets s_0, s_1 of the same length to the challenger, the challenger shares s_b for a uniformly distributed b, sends the shares of I_x to the adversary, and the adversary needs to output b.

In the j-th hybrid, we consider a similar process, where for a gate G_i, where $i = j, \ldots, 1$, if $f_i(x) = 0$, we replace the secret of gate G_i by an all-zero string of the same length. The j-th hybrid contains the shares $(\mathsf{sh}_\ell)_{x_i=1}$ generated in this process. Let pr_i be the probability that $\mathcal{A}_{\text{multislice}}$ wins the security game when it is given the shares generated in the i-th hybrid. In particular, the 0-th hybrid is the shares of I_x generated by the CSSS, thus pr_0 is the probability that $\mathcal{A}_{\text{multislice}}$ wins the security game in the CSSS. In the other extreme, in the m-th hybrid, the output is independent of the secret, since $f_m(x) = f(x) = 0$ and we replace the secret with 0. Thus, $\mathrm{pr}_m = 1/2$. We next prove that pr_{i-1} and pr_i are almost the same.

Claim 5.17. *Assume that for every $1 \leq i \leq i$ and for some constant $\alpha < 1$, the CSSS for $(n-i, n)$-slices is $t^{\alpha}(\lambda)$-secure. Furthermore, assume that $\mathcal{A}_{\text{multislice}}$ runs in time $t^{0.1\alpha}(\lambda)$. Then, $\mathrm{pr}_{i-1} - \mathrm{pr}_i \leq 2/t^{\alpha}(\lambda)$ for every $1 \leq i \leq m$.*

Proof. If $f_i(x) = 1$ then the $(i-1)$-th and the i-th hybrid are the same and the claim is trivial. We thus assume that $f_i(x) = 0$. If G_i is a leaf labeled by some variable x_j, then $x_j = 0$, i.e., the secret of this gate is not part of the shares given to I_x, and the two hybrids are equal.

Otherwise, the i-th gate is an internal gate such that $f_i(x) = 0$. Let $j_1, \ldots, j_\ell$ be the indices of the children of G_i. All these indices are smaller than i and for every j_k such that $f_{j_k}(x) = 0$ the secret of the gate G_{j_k} is the all-zero string. Since $f_i(x) = 0$, the shares of the CSSS for G_i that are not replaced are shares of a forbidden set of G_i. If G_i is either an OR, AND, or a threshold gate, the shares of the forbidden set of G are equally distributed when the secret is the all-zero string and when the secret is generated by the $(i-1)$-hybrid, i.e., also in this case the two hybrids are equal.

We are left with the interesting case where $f_i(x) = 0$ and G_i is an $(n'-i)$-slice function for some $n' < n$, $1 \leq i \leq k$. Given the adversary $\mathcal{A}_{\text{multislice}}$ trying to break the CSSS for the $(n-k:n)$-multislices, we construct the following adversary $\mathcal{A}_{\text{slice}}$ trying to break the CSSS for $(n'-i)$-slices:

Input: The security parameter 1^λ, and the $(n-k:n)$-multislice $f : \{0,1\}^n \to \{0,1\}$ and the input x s.t. $f(x) = 0$ that are generated by $\mathcal{A}_{\text{multislice}}$.

- Choose $s \in \{0,1\}$ with uniform distribution.
- Generate shares similar to the $(i-1)$-th and i-th hybrids with the secret s:
  - For $j = m, m-1, \ldots, i$ generate secrets for the gates $G_m, G_{m-1}, \ldots, G_i$ as in the $(i-1)$-th hybrid (and the CSSS).
  - Let s_0 be the secret of gate i and s_1 be the all-zero string of the same length.
  - Let y be the values of the sub-formulas of the children of G_i with input x and g be the $(n'-i)$-slice computed by G_i.
  - Send g, y, and s_0, s_1 to the Challenger of the CSSS for slices, and get the shares of I_y for the secret s_b for a uniformly distributed b.

  - For every ℓ such that $y_\ell = 0$ choose the all-zero string of the appropriate length as the ℓ-th share.
  - For $j = i-1, i-2, \ldots, 1$ generate secrets for the gates $G_{i-1}, G_{i-2}, \ldots, G_1$ as in the hybrids using the shares generated for the CSSSs for the gates $G_m, \ldots, G_i$.
  - The share of P_j are the secrets of all leaves labeled by x_j.
- Give the shares of I_x to the adversary $\mathcal{A}_{\text{multislice}}$ and get a secret $s' \in \{0,1\}$.
- If $s = s'$ output $b' = 0$, otherwise output $b' = 1$.

We start by analyzing the running time of $\mathcal{A}_{\text{slice}}$. As we assume that $\mathcal{A}_{\text{slice}}$ runs in time $t^{0.1\alpha}(\lambda)$ and it outputs a multislice f whose representation size is at least $\binom{n}{k} \geq n^{k/2}$ (since $k \leq \sqrt{n}$),

$$n^{k/2} \leq t^{0.1\alpha}(\lambda). \tag{1}$$

The running time of $\mathcal{A}_{\text{slice}}$ is dominated by the running time of $\mathcal{A}_{\text{multislice}}$ and by the running of the sharing algorithm of the CSSS for the multislice f, i.e., it is $\tilde{O}(n^{4k}) \cdot \text{poly}(\lambda) + t^{0.1\alpha}(\lambda) < t^{\alpha}(\lambda)$ (by (1)).

We are ready to bound $\text{pr}_{i-1} - \text{pr}_i$. Let b be the bit chosen by the challenger. By the $t^{\alpha}(\lambda)$-security of the CSSS for the slice functions,

$$\Pr[\mathcal{A}_{\text{slice}} \text{ wins the security game}] = \Pr[b = b'] \leq 1/2 + 1/t^{\alpha}(\lambda). \tag{2}$$

Observe that if $b = 1$ then the secret of G_i is s_1, the all-zero string; if $b = 0$ then the secret of G_i is s_0 as generated by the CSSS. In other words, if $b = 1$ the shares are generated as in the i-th hybrid, and if $b = 0$ the shares are generated as in the $(i-1)$-th hybrid. The adversary $\mathcal{A}_{\text{slice}}$ outputs $b = b'$ if

- $b = 0$ and $\mathcal{A}_{\text{multislice}}$ outputs $s = s'$, or
- $b = 1$ and $\mathcal{A}_{\text{multislice}}$ outputs $s \neq s'$.

That is, $\mathcal{A}_{\text{slice}}$ outputs $b = b'$ with probability

$$0.5\text{pr}_{i-1} + 0.5(1 - \text{pr}_i) = 0.5 + 0.5(\text{pr}_{i-1} - \text{pr}_i).$$

Thus, by (2), $\text{pr}_i - \text{pr}_{i-1} \leq 2/t^{\alpha}(\lambda)$. ■ (of Claim 5.17)

Concluding the security proof the CSSS for multislices, we assume that $\mathcal{A}_{\text{multislice}}$ runs in time $t^{0.1\alpha}(\lambda)$ and $t^{\alpha}(\lambda) > \ell_k^{10/9}$; the last assumption holds since $t^{\alpha}(\lambda) \geq k^{ck}\,\text{polylog}(n)$ for a sufficiently large c. By Claim 5.17, the probability that $\mathcal{A}_{\text{multislice}}$ wins the security game in the CSSS, i.e., pr_0, is bounded by

$$\begin{aligned} \text{pr}_0 &= (\text{pr}_0 - \text{pr}_1) + \cdots + (\text{pr}_{m-1} - \text{pr}_m) + \text{pr}_m \\ &\leq 1/2 + \frac{m}{t^{\alpha}(\lambda)} = 1/2 + \frac{O(\ell_k)}{t^{\alpha}(\lambda)} \leq \frac{1}{t^{0.1\alpha}(\lambda)}, \end{aligned}$$

since $m = O(\ell_k)$. Thus, every adversary that runs in time at most $t^{0.1\alpha}(\lambda)$ break the CSSS for the multislices with probability at most $1/t^{0.1\alpha}(\lambda)$, i.e., the CSSS is $t^{0.1\alpha}(\lambda)$-secure. □

5.5 Linear and Multi-Linear Schemes for $(n-k:n)$-Multislices

Next, we provide upper and lower bounds on the share size and information ratio for linear and multi-linear schemes. Notice that the gap between the bounds is asymptotically tight when k is constant.

Theorem 5.18 (Linear Schemes for $(n-k:n)$-Multislices). *Let $1 < k < n/\log^2 n$. Then $(n-k:n)$-multislices can be realized by a linear secret-sharing scheme with share size $\tilde{O}(k^{5k}n^{(k-1)/2})$.*

Proof. By [BP18,LVW18], the share size of linear schemes for (k,kn)-partite slices (which are equivalent to k-server CDS protocols) is at most

$$c_{\text{partite-slice}}(k,kn,1) = O\left(n^{\frac{k-1}{2}}\right).$$

By Lemma 5.1, the share size of linear schemes for dual of (k,nk)-partite slices is at most $c_{\text{partite-slice}}(k,kn,1) = O(n^{\frac{k-1}{2}})$. Thus, by Lemma 5.11, the share size of linear schemes for $(n-k:n)$-multislices is at most

$$\begin{aligned} O(c_{\text{partite-slice}}(k,kn,1)\cdot \text{polylog}(n)\cdot \ell_k) &= \tilde{O}\left(n^{\frac{k-1}{2}}\cdot k^{4k}2^k\log^{2k-1}k\right) \\ &= \tilde{O}(n^{\frac{k-1}{2}}k^{5k}). \end{aligned}$$

□

Theorem 5.19 (Multi-Linear Secret-Sharing Schemes for $(n-k:n)$-Multislices). *For $1 < k < \log n \log\log(n)$, every $(n-k:n)$-multislice f can be realized by a multi-linear secret-sharing scheme with secrets of size $2^{n^{k-1}}$ with information ratio $O(k^{5k}\log^2 n)$. For $\log n\log\log(n) < k < n/\log^2 n$, every $(n-k:n)$-multislice f can be realized by a multi-linear secret-sharing scheme with secrets of size $2^{n^{O(k)}}$ with information ratio $2^{O(k)}$.*

Proof. The construction of the multi-linear secret-sharing schemes (with long secrets) is similar to the construction of the secret-sharing scheme in Lemma 5.11. By Lemma 5.7, in order to construct a secret-sharing scheme for f it suffices to show how to realize duals of (h_i,k) hypergraphs. We realize the dual of an (h_i,k) hypergraph by the formula-based secret-sharing scheme of Lemma 5.2 using the dual of the formulas of Lemmas 5.9 and 5.10. The gates in these dual formulas are AND, OR, threshold, and dual of $(\kappa,\kappa N)$-partite slices. We use the fact that for $m > \log n$, AND, OR, and threshold functions can be realized by a secret-sharing scheme with information ratio 1. The share size required to realize the dual of $(\kappa,\kappa N)$-partite slices is the same as the share size required to realize $(\kappa,\kappa N)$-partite slices (by Lemma 5.1 – the duality of multi-linear secret-sharing schemes). We use the multi-linear k-server CDS protocols from [AA18] to realize $(\kappa,\kappa N)$-partite slices; the information ratio of the resulting scheme is $O(1)$ for long secrets of size $2^{N^{k-1}}$. Thus, by the last item in Lemma 5.2, the information ratio of multi-linear schemes for $(n-k:n)$-multislices is $O(\sum_{i=1}^{k}\ell_i)$, where ℓ_i is the size of a formula realizing (h_i,k) hypergraphs.

For $k < \log n \log\log(n)$, we use the formula of Lemma 5.9. In this case $\sum_{i=1}^{k} \ell_i = O(k^{5k} \log^2 n)$ and the secrets are of size $2^{n^{k-1}}$, as we use (k, nk)-partite slice gates. For $\log n \log\log(n) < k < n/\log^2 n$, we use the formula of Lemma 5.10. In this case $\sum_{i=1}^{k} \ell_i = 2^{O(k)}$ and the secrets are of size $2^{n^{O(k)}}$, as we use $(O(k/\log k), n^{O(\log k)})$-partite slice gates. □

In order to understand these upper bounds, we provide the following lower bounds.

Theorem 5.20. *For almost all $(n-k:n)$-multislices, the total share size in every linear secret-sharing scheme with a one-bit secret realizing these access structures is $\Omega(n^{(k-1)/2}/k^{(k+1)/2})$.*

To prove this result, we need the a result for access structures with bounded rank, from [BFMP22]. We say that an access structure has *rank* r if its minimal authorized sets are of size at most r. Theorem 5.21 is obtained by counting the number of possible matrices of a certain size for L. The same argument can be applied for a family of access structures L whose duals have rank at most r.

Theorem 5.21 ([BFMP22]). *Let L be a family of access structures with rank at most r. Then, for almost all access structures in L, the max share size for sharing a one-bit secret in a linear secret-sharing scheme is $\Omega(\sqrt{\log |L|/rn})$.*

Proof of Theorem 5.20. The dual of $(n-k:n)$-multislices are access structures whose maximal forbidden subsets are of size at most k. That is, access structures of rank $(k+1)$. The number of such access structures is greater than $2^{\binom{n}{k}}$. Applying Theorem 5.21 we have that for almost all of these access structures, the max share size for sharing a one-bit secret in a linear secret-sharing scheme is

$$\Omega\left(\sqrt{\binom{n}{k}/(k+1)n}\right) \geq \Omega\left(\sqrt{n^k/k^k(k+1)n}\right) = \Omega\left(\sqrt{n^{k-1}/k^{k+1}}\right).$$

□

Theorem 5.22. *For every k and $n > 2k$, there exist $(n-k:n)$-multislices that require information ratio $\Omega(k/\log k)$.*

Proof. This lower bound is obtained by constructing a multislice from the Csirmaz access structures [Csi97]. These access stuctures are $(\ell : n')$-multislices for an integer ℓ and $n' = \ell + 2^{\ell}$. These multislices require information ratio $\Omega(n'/\log n')$. Now choose the smallest ℓ with $k > 2^{\ell}$ and consider the Csirmaz' function f' on $\{0,1\}^{n'}$. Define $f : \{0,1\}^{n'} \times \{0,1\}^{n-n'} \to \{0,1\}$ as $f(x', x'') = f'(x') \wedge f''(x'')$, where f'' that is the AND of the last $n - n'$ variables. The information ratio for this function is at least $\Omega(n'/\log n') = \Omega(k/\log k)$. □

Remark 5.23. In Theorems 5.12, 5.16, 5.18 and 5.19, we construct schemes for $(n-k:n)$-multislices from schemes for $(nk-k, nk)$-slices or $(nk-k, nk)$-partite slices following similar black-box transformations. For $k \leq \log n(\log\log n)^2$, the share size of the resulting schemes are, approximately, $O(k^{5k})$ times the share size of the scheme for $(n-k)$-slices. When $k = O(\log n/(\log\log n)^2)$, this term is $n^{o(1)}$. For $k \geq \log n(\log\log n)^2$, the share size is $2^{O(k)}$.

Next, we detail some particular cases. In Theorem 5.12, if k is constant, the share size is $n^{1+o(1)}$, while the best previous upper bound was $O(\binom{n-1}{k-1})$. In Theorem 5.18, we provide the first computational scheme for multislices assuming OWFs. Based on the harness assumption for RSA, the scheme in [ABI+23b] for $(n-k : n)$-multislices has shares of size polynomial in $k \log n$ and the security parameter. In our case, for constant k, the share size is $O(\lambda \log^3 n)$. Analogously to the perfect case (see Fig. 8), the share size of our CSSSs improves the previous upper bounds for $k = O(n^{1/6})$.

Comparing Theorem 5.16 and Theorem 5.20, we see that our linear schemes for $(n-k : n)$-multislices are asymptotically optimal for constant k. Our multi-linear schemes for $(n-k : n)$-multislices have constant information ratio for constant k. For linear and multi-linear schemes, the best previous bound for small k was also $O(\binom{n-1}{k-1})$. Notice that for $k = 2, 3$, the schemes with the smallest share size are the linear ones in Theorem 5.16 (the bound of Theorem 5.12 is worse). The existence of better non-linear schemes is an open problem.

In the case of linear schemes, the best scheme for k between $n/2$ and $n/8$ has share size $\tilde{O}(2^{(n+k)/2})$ [AN21]. For $k = O(n^{1/12})$, the best scheme is the one in Theorem 5.16. For intermediate values of k, the best bound is $O(\binom{n-1}{k-1})$. Regarding the information ratio of multi-linear schemes, our schemes are better than the previous ones for $k = O(n^{1/6})$, analogously to the perfect case (Fig. 8).

Acknowledgments. We thank Benny Applebaum and Eliran Kachlon for valuable comments on earlier drafts of this paper. The first author is supported by the ISF grant 391/21 and by the ERC grant 742754 (project NTSC). The second author is supported by the grant 2021SGR 00115 from the Government of Catalonia, the project ACITHEC PID2021-124928NB-I00 from the Government of Spain, and the project HERMES funded by the European Union NextGenerationEU/PRTR via INCIBE. The third author is supported by the ISF grant 391/21 and by the Frankel center for computer science. The forth author is supported by ISF grant no. 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

AA18. Benny Applebaum and Barak Arkis. On the power of amortization in secret sharing: d-uniform secret sharing and CDS with constant information rate. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018*, volume 11239 of *LNCS*, pages 317–344. Springer-Verlag, 2018.

ABF$^+$19. Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019*, volume 11478 of *LNCS*, pages 441–471. Springer-Verlag, 2019.

ABI$^+$23a. Damiano Abram, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Varun Narayanan. Cryptography from planted graphs: Security with logarithmic-size messages. In Guy N. Rothblum and Hoeteck Wee, editors, *TCC 2023*, volume 14369 of *LNCS*, pages 286–315, 2023.

ABI$^+$23b. Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In *STOC 2023*, pages 1553–1566. ACM, 2023.

ABI$^+$23c. Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. Technical Report 2023/955, Cryptology ePrint Archive, 2023.

ABN$^+$22. Benny Applebaum, Amos Beimel, Oded Nir, Naty Peter, and Toniann Pitassi. Secret sharing, slice formulas, and monotone real circuits. In *ITCS 2022*, volume 215 of *LIPIcs*, pages 8:1–8:23, 2022.

ABNP20. Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In *STOC 2020*, pages 280–293. ACM, 2020.

ADH17. Varunya Attasena, Jérôme Darmont, and Nouria Harbi. Secret sharing for cloud data security: a survey. *The VLDB Journal*, 26(5):657–681, 2017.

AN21. Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1.5^{n}. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021*, volume 12827 of *LNCS*, pages 627–655. Springer, 2021.

BC94. Amos Beimel and Benny Chor. Universally ideal secret sharing schemes. *IEEE Trans. on Information Theory*, 40(3):786–794, 1994.

Bei11. Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology – Third International Workshop, IWCC 2011*, volume 6639 of *LNCS*, pages 11–46. Springer-Verlag, 2011.

Bei23. Amos Beimel. Lower bounds for secret-sharing schemes for k-hypergraphs. In *ITC 2023*, volume 267 of *LIPIcs*, pages 16:1–16:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

Ber82. S. Berkowitz. On some relationships between monotone and nonmonotone circuit complexity. Technical report, Department of Computer Science, University of Toronto, 1982.

BF20a. Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. *IACR Cryptol. ePrint Arch.*, 2020:664, 2020.

BF20b. Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. In *TCC 2020*, volume 12552 of *LNCS*, pages 499–529, 2020.

BFLN24. Amos Beimel, Oriol Farràs, Or Lasri, and Oded Nir. Secret-sharing schemes for high slices. In *TCC*, 2024.

BFMP22. Amos Beimel, Oriol Farràs, Yuval Mintz, and Naty Peter. Linear secret-sharing schemes for forbidden graph access structures. *IEEE-TIT*, 68(3):2083–2100, 2022.

BGK16. Andrej Bogdanov, Siyao Guo, and Ilan Komargodski. Threshold secret sharing requires a linear size alphabet. In *TCC 2016*, volume 9986 of *LNCS*, pages 471–484, 2016.

BIKK14. Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 317–342. Springer-Verlag, 2014.

BKN18. Amos Beimel, Eyal Kushilevitz, and Pnina Nissim. The complexity of multiparty PSM protocols and related models. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10821 of *LNCS*, pages 287–318. Springer-Verlag, 2018.

BL88. Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shaffi Goldwasser, editor, *CRYPTO '88*, volume 403 of *LNCS*, pages 27–35. Springer-Verlag, 1988.

Bla79. George Robert Blakley. Safeguarding cryptographic keys. In *Proc. of the 1979 AFIPS National Computer Conference*, volume 48 of *AFIPS Conference proceedings*, pages 313–317. AFIPS Press, 1979.

Bog23. Andrej Bogdanov. Csirmaz's duality conjecture and threshold secret sharing. In Kai-Min Chung, editor, *ITC*, volume 267 of *LIPIcs*, pages 3:1–3:6, 2023.

BP18. Amos Beimel and Naty Peter. Optimal linear multiparty conditional disclosure of secrets protocols. In Thomas Peyrin and Steven D. Galbraith, editors, *ASIACRYPT 2018*, volume 11274 of *LNCS*, pages 332–362. Springer, 2018.

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

CK93. Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *J. of Cryptology*, 6(2):87–96, 1993.

Csi97. László Csirmaz. The size of a share must be large. *J. of Cryptology*, 10(4):223–231, 1997.

Csi20. László Csirmaz. Secret sharing and duality. *J. Math. Cryptol.*, 15(1):157–173, 2020.

CSNN24. Arup Kumar Chattopadhyay, Sanchita Saha, Amitava Nag, and Sukumar Nandi. Secret sharing: A comprehensive survey, taxonomy and applications. *Computer Science Review*, 51:100608, 2024.

EP97. Paul Erdös and László Pyber. Covering a graph by complete bipartite graphs. *Discrete Mathematics*, 170(1–3):249–251, 1997.

Feh98. Serge Fehr. Span programs over rings and how to share a secret from a module. Master's thesis, ETH Zurich, 1998.

FHKP17. Oriol Farràs, Torben Brandt Hansen, Tarik Kaced, and Carles Padró. On the information ratio of non-perfect secret sharing schemes. *Algorithmica*, 79(4):987–1013, 2017.

Gál95. Anna Gál. *Combinatorial Methods in Boolean Function Complexity.* PhD thesis, U. of Chicago, 1995.

GIKM00. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and System Sciences*, 60(3):592–629, 2000.

GKW15. Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption.

In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9216 of *LNCS*, pages 485–502. Springer-Verlag, 2015.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Construction of a pseudo-random generator from any one-way function. *SIAM J. on Computing*, 28(4):1364–1396, 1999.

ISN87. Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Globecom 87*, pages 99–102, 1987. Journal version: Multiple assignment scheme for sharing secret. *J. of Cryptology*, 6(1), 15-20, 1993.

KGH83. Ehud D. Karnin, Jonathan W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Trans. on Information Theory*, 29(1):35–41, 1983.

KN90. Joe Kilian and Noam Nisan, 1990. Unpublished result.

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

Kra94. Hugo Krawczyk. Secret sharing made short. In *CRYPTO '93*, volume 773 of *LNCS*, pages 136–146. Springer-Verlag, 1994.

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

LS20. Kasper Green Larsen and Mark Simkin. Secret sharing lower bound: Either reconstruction is hard or shares are long. In *SCN 2020*, volume 12238 of *LNCS*, pages 566–578. Springer, 2020.

LV18. Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *50th STOC*, pages 699–708, 2018.

LVW18. Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 567–596. Springer-Verlag, 2018.

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

SS97. Hung-Min Sun and Shiuh-Pyng Shieh. Secret sharing in graph-based prohibited structures. In *INFOCOM '97*, pages 718–724, 1997.

VNS$^+$03. V. Vinod, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *Indocrypt 2003*, volume 2904 of *LNCS*, pages 162–176. Springer-Verlag, 2003.

Yao89. Andrew C. Yao. Unpublished manuscript, 1989. Presented at Oberwolfach and DIMACS workshops.

A Proof of Lemma 5.10 – A Formula for (h_i, k)-Hypergraphs

In this section, we prove Lemma 5.10, i.e., we show an improved formula of size $2^{O(k)}$ for (h_i, k)-hypergraphs when $k \geq \log n(\log\log n)$. The formula is an adaptation of a construction of [ABNP20], taking different parameters (in [ABNP20], they considered $((1/2-\delta)n : (1/2+\delta)n)$-multislice for some constant δ). For this construction, we define partite hypergraphs and robust partite hypergraphs.

Definition A.1 (Partite Hypergraphs and Robust Partite Hypergraphs). *A (k, kN)-hypergraph $h : [kN] \to \{0,1\}$ is* (k, kN)-partite *if for every minterm*

x of h (of weight k) and for every $j \in [k]$

$$|\{\ell : x_\ell = 1\} \cap \{(j-1)N+1, \dots, jN\}| = 1,$$

that is, the variables are partitioned to k subsets of size N and in each minterm of weight k there is exactly one satisfied variable in each subset.

We say that a function $h : [kN] \to \{0,1\}$ is a t-robust (k, kN)-partite hypergraph *if there is a (k, kN)-partite hypergraph h' such that*

$$h(x) = h'(x) \vee \bigvee_{j=1}^{k} \mathrm{TR}_{t+1,N}(x_{(j-1)k+1}, \dots, x_{jk}),$$

i.e., $h(x) = 1$ if $h'(x) = 1$ or if it contains at least $t+1$ satisfied variables from some subset $\{(j-1)k+1, \dots, jk\}$.

Observe that t-robust (k, kN)-partite hypergraphs are basically equivalent to k-server robust CDS protocols (defined in [ABNP20]) and can be realized by a secret-sharing scheme with share size $\log N \cdot k^{O(k)} \cdot (t \log^2 t)^k 2^{O(\sqrt{k \log N}) \log\log N)}$ [ABNP20]; in particular, they can be computed by a small monotone formula with the following properties.

Theorem A.2 (Implicit in [ABNP20], Explicit in [ABN$^+$22]). *Every t-robust (k, kN)-partite hypergraph can be computed by a constant depth formula of size $\log N \cdot k^{O(k)} \cdot (t \log^2 t)^k$ with AND,OR, and (k, kN)-slices gates.*

Proof of Lemma 5.10. The construction has two stages. We first construct a formula that accepts every minterm (of weight i) with probability at least $3/4$ and rejects every maxterm (of weight k) with probability at least $3/4$; the construction is randomized and the probability is over the randomness of the construction. We then use a few copies of the construction from the first stage to reduce the error; by a simple union bound we prove that there exists a scheme that accepts all minterms and rejects all maxterms. In the proof we construct a formula of size $2^{O(k)}$, we do not try to optimize the constant in the exponent. Furthermore, for various values of i the formula size can be reduced; however, as we are interested in the maximum size over all $1 \leq i \leq k$, and we do not try to improve the formula size for such values of i. The worst case is roughly when $i = k/2$.

First Stage. Let $k' \leq k$ be a parameter to be fixed later. We randomly partition $[n]$ to k' sets $C_1, \dots, C_{k'}$, where $\Pr[\ell \in C_j] = 1/k'$ for every $\ell \in [n], j \in [k']$. Using this partition, we define a t-robust $(k', k'N)$-hypergraph that will be used to compute the (h_i, k)-hypergraph.

Fix a set B of size k and $j \in [k']$, w.l.o.g., let $B = \{1, \dots, k\}$. The expected value of $|B \cap C_j|$ is $|B|/k' = k/k'$. By the Chernoff bound, the size of $B \cap C_j$ is close to the expected value with high probability. Recall the following Chernoff bound. Let $0 < p < 1$ and $X_1, \dots, X_k$ be independent binary random variables such that $\Pr[X_\ell = 1] = p$ for every $\ell \in [k]$ and let $X = \sum_{\ell=1}^{k} X_\ell$. Then, denoting $\mu = \mathbb{E}[X] = kp$, for every $0 < \delta < 1$ we have

$$\Pr[X > (1+\delta)\mu] \leq e^{-\mu\delta^2/3}.$$

Thus, letting $X_\ell = 1$ if $\ell \in C_j$,

$$\begin{aligned}\Pr\left[|B \cap C_j| \geq \frac{k}{k'} + \sqrt{\frac{3k\ln(4k')}{k'}}\right] &= \Pr\left[X \geq \left(1 + \sqrt{\frac{3k'\ln(4k')}{k}}\right)\frac{k}{k'}\right] \\ &\leq e^{-\frac{k}{k'}\cdot\frac{3k'\ln(4k')}{k}/3} = \frac{1}{4k'}.\end{aligned} \tag{3}$$

We will take k' such that $\sqrt{3k\ln(4k')/k'} \leq k/k'$, e.g.,

$$k' \leq \frac{k}{3\log k}. \tag{4}$$

Notice that in this case $\delta \leq 1$ and we can apply the Chernoff bound. By the union bound,

$$\begin{aligned}&\Pr\left[\forall_{j\in[k']}|B \cap C_j| \leq \frac{2k}{k'}\right] \\ &\qquad \geq \Pr\left[\forall_{j\in[k']}|B \cap C_j| \leq \frac{k}{k'} + \sqrt{\frac{3k\ln(4k')}{k'}}\right] \geq 3/4.\end{aligned} \tag{5}$$

The same conclusions holds for a set B such that $|B| \leq k$. In particular, for every minterm (of weight i, where $i \leq k$) with probability at least $3/4$, the set $B = \{\ell : x_\ell = 1\}$ contains at most $2k/k'$ elements from each set C_j.

The above fact motivates the definition of the following $t = 2^{2k/k'}$-robust $(k', k'N)$-hypergraph h (the choice of t will be explained below), where

1. $N = \binom{n}{\leq 2k/k'} \leq n^{2k/k'}$.
2. For every j we index the first $\binom{|C_j|}{\leq 2k/k'}$ variables in the j-th part (i.e., $x_{(j-1)N+1}, \ldots, x_{jN}$) by subsets of C_j of size at most $2k/k'$,
3. For a minterm $x \in \{0,1\}^n$ of h_i (of weight i), let $A_j = \{\ell \in C_j : x_\ell = 1\}$ for $j \in [k']$ (i.e., $A_1 \cup \cdots \cup A_{k'}$ are the coordinates in which x is one). If $|A_j| \leq 2k/k'$ for every $j \in [k']$, then for the input $y \in \{0,1\}^{kN}$ such that $y_{A_j} = 1$ for every $j \in [k']$ and all other coordinates are zero we define $h(y) = 1$.
4. For every y of weight k' such that $f(y)$ is not defined in item 3, we define $h(y) = 0$.
5. For every y of weight greater than k, we define $h(y) = 1$ if and only if
   - There exists y_0 of weight k such that $h(y_0) = 1$ and $y_0 \leq y$, or
   - There exists $j \in [k']$ such that

$$|\{\ell : y_\ell = 1\} \cap \{(j-1)N+1, \ldots, jN\}| > t.$$

Example A.3. Let $k = 12, k' = 3$, and $n = 100$. Let $C_1 = \{1, \dots, 40\}$, $C_2 = \{41, \dots, 57\}$, and $C_3 = \{58, \dots, 100\}$ be the random partition. Consider the input x, where $x_1 = \cdots = x_7 = 1$, $x_{45} = x_{46} = 1$, and $x_{91} = x_{92} = x_{93} = 1$ and assume that $h_{12}(x) = 1$. Then, $h(y) = 1$ for the input y of weight 3, where $y_{\{1,\dots,7\}} = y_{\{45,46\}} = y_{\{91,92,93\}} = 1$ and all other coordinates are 0.

Furthermore, consider the input x, where $x_1 = \cdots = x_9 = 1$, $x_{45} = x_{46} = 1$, and $x_{91} = 1$. Since $|\{1, \dots, 9\} \cap C_1| = 9 > 2k/k'$, there is no input y corresponding to x.

We construct the formula F for h_i by constructing the formula for h guaranteed by Theorem A.2, where we replace each input wire of h indexed by a non-empty set A by an a gate $\text{AND}((x_\ell)_{\ell \in A})$, assign 1 to the input wire labeled by an empty set, and assign 0 to the input wires that are not indexed by a set. We next analyze the properties of the construction.

Let x be a minterm of weight i of h_i and let $A_j = \{\ell : x_\ell = 1\} \cap C_j$ for $j \in [k']$. We claim that if each A_j is of size at most $2k/k'$, which occurs with probability at least 3/4 by (5), then $F(x) = 1$. Let y be such that $y_{A_j} = 1$ for every $j \in [k']$ and all other coordinates are zero. Since all A_j's are small, y is properly defined and $h(y) = 1$ by item 3 of the definition of h. Furthermore, $\text{AND}((x_\ell)_{\ell \in A_j})$ for every $j \in [k']$. Thus, $F(x) = 1$.

Let x be a maxterm of the (h_i, k)-hypergraph, in particular, the weight of x is k. a Let $A_j = \{\ell : x_\ell = 1\} \cap C_j$ for $j \in [k']$. We claim if each A_j is of size at most $2k/k'$, which by (5) occurs with probability at least 3/4, then $F(x) = 0$. By the construction of F, the input wires of h that are satisfied by x are exactly the wires that are indexed by a set A such that $A \subseteq A_j$ for some $j \in [k']$; let y be the input to h such that $y_A = 1$ if and only if $A \subseteq A_j$. In the j'th part of y, there are at most $2^{|A_j|} \leq 2^{2k/k'}$ inputs wires of h that are satisfied by x. For every $x' \leq x$, it must be that $h_i(x') = 0$ (by the monotonicity of h_i). Thus, $h(y') = 0$ for every $y' \leq y$ of weight k'. By Item 5 of the definition of h, in particular, by the fact that h is $2^{2k/k'}$-robust, $h(y) = 0$ and $F(x) = 0$.

Second Stage. We take $u = O(k \log n)$ copies $F_1, \dots, F_u$ of the construction for F of the first stage; in each copy we choose the random partition independently. The final construction is $G(x) = \text{TR}_{u/2,u}(F_1(x), \dots, F_u(x)) \vee \text{TR}_{k+1,n}(x)$, that is, the formula returns 1 if at least $u/2$ of the copies of F output 1 or if the input has weight at least $k + 1$. Let x be an input of weight i such that $h_i(x) = 1$. By a simple Chernoff bound, more than half of the copies of F output 1 with probability at least $1 - 1/4n^k$; in this case $G(x) = 1$. Similarly, let x be an input of weight k such that $h_i(x) = 0$. By a simple Chernoff bound, more than half of the copies of F output 0 with probability at least $1 - 1/4n^k$; in this case $G(x) = 0$. By the union bound, with probability at least half, G outputs the correct value for every minterm and maxterm. Thus, there are u partitions, such that G with $F_1, \dots, F_u$ induced by these partitions computes the (h_i, k)-hypergraph.

Formula Size. It remains to analyze the size (i.e., number of gates) of G. We use the formula of Theorem A.2 for a t-robust $(k', k'N)$-partite hypergraph, where

$N = \binom{n}{2k/k'} \leq n^{2k/k'}$ and $t = 2^{2k/k'}$. The size of the formula for $k \geq \log n$ is

$$\begin{aligned} \log N \cdot k'^{O(k')} \cdot (t \log^2 t)^{k'} &= O\left(k/k' \log n \cdot 2^{O(k' \log k')} \cdot (2^{2k/k'}(2k/k')^2)^{k'}\right) \\ &= O\left(2^{O(k' \log k' + k)}\right). \end{aligned} \tag{6}$$

In F, each leaf of the formula is replaced by an AND gate and the fan-in of each gate in the formula of Theorem A.2 is at most $k'N$. Thus, the size of the formula F is $O(k'N \cdot 2^{O(k+k' \log k')}) = 2^{O(k+k' \log k' + k \log n/k')}$. Taking $k' = k/3 \log k$ (as required in (4)), we obtain that the size of the formula F is $2^{O(k+\log k \log n)}$. For $k \geq \log n \log \log n$, the size of F is $2^{O(k)}$. As G contains $O(k \log n)$ copies of F, its size is also $2^{O(k)}$. □