Electronic Colloquium on Computational Complexity, Report No. 6 (2022)

Secret Sharing, Slice Formulas, and Monotone Real Circuits[*]

Benny Applebaum[†] Amos Beimel[‡] Oded Nir[*] Naty Peter[*] Toniann Pitassi[§]

January 12, 2022

Abstract

A secret-sharing scheme allows to distribute a secret s among n parties such that only some predefined "authorized" sets of parties can reconstruct the secret, and all other "unauthorized" sets learn nothing about s. For over 30 years, it was known that any (monotone) collection of authorized sets can be realized by a secret-sharing scheme whose shares are of size $2^{n-o(n)}$ and until recently no better scheme was known. In a recent breakthrough, Liu and Vaikuntanathan (STOC 2018) have reduced the share size to $2^{0.994n+o(n)}$, and this was further improved by several follow-ups accumulating in an upper bound of $1.5^{n+o(n)}$ (Applebaum and Nir, CRYPTO 2021). Following these advances, it is natural to ask whether these new approaches can lead to a truly sub-exponential upper-bound of $2^{n^{1-\varepsilon}}$ for some constant $\varepsilon > 0$, or even all the way down to polynomial upper-bounds.

In this paper, we relate this question to the complexity of computing monotone Boolean functions by monotone real circuits (MRCs) – a computational model that was introduced by Pudlák (J. Symb. Log., 1997) in the context of proof complexity. We introduce a new notion of "separable" MRCs that lies between monotone real circuits and monotone real formulas (MRFs). As our main results, we show that recent constructions of general secret-sharing schemes implicitly give rise to separable MRCs for general monotone functions of similar complexity, and that some monotone functions (in monotone NP) cannot be computed by sub-exponential size separable MRCs. Interestingly, it seems that proving similar lower-bounds for general MRCs is beyond the reach of current techniques.

We use this connection to obtain lower-bounds against a natural family of secret-sharing schemes, as well as new non-trivial upper-bounds for MRCs. Specifically, we conclude that recent approaches for secret-sharing schemes cannot achieve sub-exponential share size and that every monotone function can be realized by an MRC (or even MRF) of complexity $1.5^{n+o(n)}$. To the best of our knowledge, this is the first improvement over the trivial $2^{n-o(n)}$ upper-bound. Along the way, we show that the recent constructions of general secret-sharing schemes implicitly give rise to Boolean formulas over slice functions and prove that such formulas can be simulated by separable MRCs of similar size. On a conceptual level, our paper continues the rich line of study that relates the share size of secret-sharing schemes to monotone complexity measures.

[*]A preliminary version has appeared in the proceedings of ITCS2022.

[†]Tel-Aviv University, Tel-Aviv, Israel, {bennyap,odednir,natypeter}@mail.tau.ac.il. Supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under Grant Agreement No. 639813 ERC-CLC, and by the Israel Science Foundation grant no. 2805/21.

[‡]Ben-Gurion University, Be'er-Sheva, Israel, amos.beimel@gmail.com. Supported by ERC grant 742754 (project NTSC), Israel Science Foundation grant no. 391/21, and a grant from the Cyber Security Research Center at Ben-Gurion University.

[§]University of Toronto and Columbia University, Toronto, Canada and New York, NY, USA, tonipitassi@gmail.com. Supported by NSERC, IAS School of Mathematics, and NSF Grant CCF-1900-460.

ISSN 1433-8092

1 Introduction

Secret-sharing schemes were originally presented by Shamir and Blakley [48, 12] at 1979, and since then have become a central cryptographic tool with a wide range of applications including secure multiparty computation protocols [9, 17], threshold cryptography [21], access control [40], attribute-based encryption [27, 55], and oblivious transfer [49, 52]. From a technical point of view, secret-sharing schemes can be viewed as a distributed analog of encryption. That is, given a secret message s the goal is to "split" it to n shares, $s_1, \ldots, s_n$ and store each share on a different device ("party") so that the secret can be recovered given "sufficiently many" different shares, whereas a "small" coalition of parties should not be able to learn anything about the secret in an information-theoretic sense. (See Definition 2.4 for a formal definition of secret-sharing schemes.)

More formally, in its general form [30], the problem is parameterized by a monotone function $f : \{0,1\}^n \to \{0,1\}$ that specifies which coalitions should be able to recover the secret: A coalition A is authorized if its characteristic vector x_A is accepted by f, and is unauthorized otherwise.[1] For example, in the canonical case of *threshold secret-sharing* the function f is a threshold function that accepts all the strings whose Hamming weight exceeds a certain threshold. For this case, Shamir's polynomial-based scheme [48] provides a solution whose complexity, measured as the *total share-size* $\sum_i |s_i|$, is quasi-linear, $O(n \log n)$, in the number of parties n.

The complexity of general secret-sharing schemes. Determining the share size of secret-sharing schemes realizing general monotone functions is a basic, well-known, open problem in information-theoretic cryptography. For almost 30 years, since the pioneering work of Ito et al. [30], all known upper-bounds on the secret-sharing cost of f (measured as the best achievable share-size) have been tightly related to the computational complexity of f measured under various computational models such as monotone formula size and monotone span-program size [10, 34, 11]. Consequently, when f is taken to be a worst n-bit monotone function, these constructions lead to exponential upper-bounds of $2^{n(1-o(1))}$.

In the past few years, the seemingly tight correspondence between computational complexity and secret-sharing complexity was challenged. In a breakthrough result, Liu and Vaikuntanathan [38] (hereafter referred to as LV) showed, for the first time, that it is possible to construct secret-sharing schemes in which the total share size is $2^{cn+o(n)}$, for some constant $c < 1$. This shows that the secret-sharing complexity of worst-case monotone functions is significantly smaller than their computational complexity, which is known to be $2^{n(1-o(1))}$, even with respect to liberal models such as Boolean circuits. The latter bound can be proved via a standard counting argument [45], see, for example, [32, Chapter 1]. While the original LV result achieved an exponent of $c \approx 0.994$, subsequent works [2, 3, 4] have shown that the secret-sharing complexity can be significantly improved culminating in an upper bound of $1.5^{n+o(n)}$ [4]. Following these advances, it is natural to ask how much additional progress can be made using these new tools. Specifically,

> Can we use "LV-based techniques" to obtain general secret-sharing schemes with truly sub-exponential upper-bound of $2^{n^{1-\varepsilon}}$ for some constant $\varepsilon > 0$?

1.1 Our Results

Formulas over slices. To answer the above question, we introduce a new natural monotone complexity measure. For a monotone function f, denote by $\mathsf{FS}(f)$ the size of the smallest *formula over slices* (FOS)

[1]Monotonicity here means that for any $A \subset B$ it holds that $f(x_A) \leq f(x_B)$. It is not hard to see that a non-monotone function does not admit a secret-sharing scheme, and therefore this requirement is necessary.

that computes f, where a formula over slices is a formula such that each gate computes some (k, ℓ)-*slice* function $g : \{0,1\}^\ell \to \{0,1\}$ that takes arbitrary values on inputs of Hamming weight k, rejects lighter inputs, and accepts heavier inputs. The values of k (the weight of the gate) and ℓ (the fan-in of the gate) can vary between different gates in the formula and are allowed to be arbitrarily large. Since AND/OR gates are also slice functions, $\mathsf{FS}(f)$ is upper-bounded by the size of the (standard) monotone formula that computes f. Of course, the FOS model is much stronger. The number of $(n/2, n)$-slices is $2^{\binom{n}{n/2}}$, and so, by counting, even a single slice gate cannot be simulated by a small (e.g., sub-exponential) monotone circuit.

Recent secret-sharing yield formulas over slices. In Section 5, we show that all known non-linear constructions [38, 2, 3, 4, 8] of secret-sharing schemes with non-trivial share size (2^{cn} for a constant $c < 1$) give rise to FOS of similar size.[2] That is, we show that these constructions implicitly take the following route: (1) Realize f via a 2^{cn}-size formula F over a sub-family of slice functions that have a relatively cheap secret-sharing implementation (a.k.a. CDS protocols) [39]; (2) Use a generic transformation from formulas to secret-sharing (see Appendix B) that yields a secret-sharing scheme with share size 2^{cn}. While [38] already observed that their scheme can be described under the above framework, this observation is less apparent for some of the subsequent constructions, e.g., [3, 4, 8].[3] Specifically, based on [3], we prove the following theorem.

Theorem 1.1. *Every monotone function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a constant-depth FOS F of size $1.5^{n+o(n)} = 2^{0.585n+o(n)}$.*

From FOS to monotone real circuits (MRCs). Getting back to our motivating question, we ask whether it is possible to prove a sub-exponential upper-bound on $\mathsf{FS}(f)$ for a general n-bit monotone function. We cope with this question by turning FOS into *monotone real circuits* (MRCs) [43]. MRCs generalize the standard notion of monotone Boolean circuits by making use of fan-in 2 *monotone real gates* that compute arbitrary real-valued operators $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that are monotone over the reals, i.e., for every $x \le x'$ and $y \le y'$, it holds that $g(x, y) \le g(x', y')$. A beautiful result of Rosenbloom [46] shows that any slice function SL : $\{0,1\}^n \to \{0,1\}$ can be computed by a read-twice monotone real formula (MRF) F_{SL} of size $O(n)$.[4] Consequently, any FOS F can be converted into an MRC F' of similar size. The resulting circuit has many gates of fan-out 2 (originating from the read-twice inputs of F_{SL}) and so it is not an MRF. (Indeed, we do not know whether FOS can be simulated by MRFs with polynomial overhead.) This is unfortunate since for MRCs the best known lower-bounds are sub-exponential 2^{n^ε} for constants $\varepsilon < 1$ (based on extensions of Razborov's approximation method [44, 43]). No better lower bounds are known for MRCs (even for implicit functions). For MRFs one can hope to prove stronger lower-bound via communication complexity methods [33, 36].

Separable MRCs. We bypass the above problem by observing that the circuit F', which is obtained by computing a formula F over Rosenbloom's formulas F_{SL}, has small *separators*. Roughly speaking, every rooted sub-circuit F'_0 of F' can be "broken" to $k = O(1)$ sub-circuits each containing at most α-fraction

[2]There are some linear constructions that are not captured by this framework (e.g., in the appendix of [3]), however for such linear constructions an exponential lower-bound of $2^{n/2}$ is known [5].

[3]The latter works develop "immunization" tools that allow to take simple secret-sharing schemes and turn them into "robust" schemes that can be employed several times while re-using the same randomness. Somewhat surprisingly, these tools can be eventually translated to FOS constructions; See Section 5.

[4]A monotone real circuit computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every binary input $x \in \{0,1\}^n$ the circuit outputs the Boolean value $f(x)$. Note that the intermediate values induced on internal wires may not be binary.

of the nodes of F'_0 for some constant $\alpha < 1$. This notion of "separable circuits" generalizes the notion of formulas (for which $k = 1$ and $\alpha = 2/3$). Indeed, in the context of Boolean circuits, it is known that separability can be used to "balance" the circuit and turn it into a formula of comparable size [23]. While we do not know how to prove a similar result for separable MRCs, we can show that formula lower-bound techniques extend to this case as well. Specifically, we prove that the size of separable MRCs is exponential in the randomized communication complexity of the corresponding KW-game, extending the result of Krajíček [36] that was originally proved for MRFs. Together with a randomized communication complexity lower bound of Göös and Pitassi [26], we derive the following result. (See Section 3.)

Theorem 1.2. *There exists a function in monotone NP that requires size $2^{\Omega(n/\log^2 n)}$ formulas over slice gates. Moreover, this holds even for formulas that use both slice gates and monotone real gates.*

We do not know whether logarithmic terms in the exponents can be shaved, but we observe that if the bound is tight and the fan-in of the slice gates is bounded by a polynomial in n, then one can obtain an interesting improvement on the *rate* of secret-sharing schemes for very long secrets. In fact, such an improvement can be obtained even if the upper-bound is $2^{o(n/\log n)}$ and even if only the *weight* of the slice gates is restricted to $\mathrm{poly}(n)$ but the fan-in may be arbitrary. (See Section 6.)

Theorem 1.3. *Suppose that the function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a FOS of size $2^{o(n/\log n)}$ over slice functions of weight bounded by $\mathrm{poly}(n)$. Then, for sufficiently long secret s, the function f can be realized with share size $2^{o(n)} \cdot |s|$.*

We mention that currently we do not have non-trivial upper-bounds on the rate of worst-case secret-sharing (even for very long secrets) apart from the ones that follow from the case of single-bit shares (e.g., $1.5^{n+o(n)} \cdot |s|$).

Moving back to upper-bounds, we observe that existing secret-sharing schemes also give rise to non-trivial MRCs and even MRFs. In particular, by plugging in Rosenbloom's construction in the FOS obtained by Theorem 1.1, and by exploiting the fact that the depth of the FOS of Theorem 1.1 is constant, we derive the following upper-bound on the worst-case complexity of MRFs for n-bit functions (also known as the *Shannon function* [32] of MRFs).

Corollary 1.4. *Every monotone function $f : \{0,1\}^n \to \{0,1\}$ can be computed by an MRF F of size $1.5^{n+o(n)} = 2^{0.585n+o(n)}$.*

To the best of our knowledge, this is the first non-trivial improvement over the naive $2^{n-o(n)}$ bound, even for the case of MRCs. An even more dramatic improvement can be obtained for "typical" monotone functions based on the results of Beimel and Farràs [7]. Specifically, all but $o(1)$-fraction of all n-bit monotone functions can be realized by an MRF of size $2^{\tilde{O}(\sqrt{n})}$. (See Section 5.1.)

Secret-sharing vs. MRCs. While the worst-case upper bounds for MRCs and secret-sharing schemes are currently equal, we observe that for concrete functions secret-sharing complexity and MRC size can be separated. Specifically, in Section 4, we show that secret-sharing complexity can be super-polynomially cheaper than MRC size and exponentially cheaper than FOS and MRF sizes. On the other direction, we derive an almost quadratic separation, that is, we construct an MRF of size $O(n)$ for an explicit function that, by [19], requires total share size $O(n^2/\log n)$; this is the best possible given that existing secret-sharing lower-bounds [19]. We note that there are concrete functions for which the share size of the best known secret-sharing scheme is super-polynomially larger than the MRC size. Most notably, the best secret-sharing construction of $(n/2, n)$-slices has share size of $2^{\tilde{O}(\sqrt{n})}$ [39, 38], whereas such functions can be realized by a single slice gate, i.e., a linear size MRC (or even MRF). We further present a $2^{\Omega(n)}$ gap for the case of uniformly chosen DNFs of $\Omega(n)$ width. We prove that the same gap also exists, perhaps more surprisingly, between FOS and secret-sharing. Along the way, we prove that MRCs are closed under duality – an interesting property that may be useful elsewhere. (See Appendix A.)

Conclusion and open questions. Our work continues the rich line of study that relates the share size of secret-sharing schemes to monotone complexity measures. We import lower-bounds from the computational complexity world to the domain of secret-sharing schemes and use recent constructions of secret-sharing schemes to obtain new algorithmic results for several monotone computational models. Our results highlight several interesting open questions in both domains. We list some of them here.

First, it will be interesting to better understand the power of formulas over slices (possibly with some bound on the fan-in). What is the relation between such formulas and monotone real formulas? As far as we know these two models may be incomparable. Also, we know how to balance FOS, so is it possible to balance MRFs as well? On the secret-sharing front, it is natural to ask whether one can beat the FOS lower bound. One potential route is to replace some of the existing steps with "non-FOS-able realizations". Most notably, as mentioned in Footnote 3, one of the important ingredients in recent constructions is some form of "robust" secret-sharing for simple functions (a.k.a. robust CDS protocols) [3]. While we showed that the main instantiations of this primitive can be cast as FOS, one may still hope to find other realizations that do not have this feature. Indeed, some linear and quadratic realizations of this primitive [3, 8] do not seem to have a "FOS interpretation", though these constructions are currently too expensive to be useful.

1.2 Other Related Work

Monotone real circuits. Monotone real circuits were defined by Pudlák [43], whose motivation was proof complexity applications, i.e., proving lower bounds for cutting planes proofs. Exponential lower bounds for monotone real circuits were obtained in [43, 28, 51, 31, 24]. Specifically, for a function $f : \{0,1\}^n \to \{0,1\}$ the best lower bound is $2^{\tilde{\Omega}(n^{1/3})}$ [51] (this function is only partially explicit). For an explicit function the best known lower bound is $2^{\Omega(n^{1/4}\sqrt{\ln n})}$ [31, 32]. Hrubeš and Pudlák [29] proved that if an n-bit function can be computed by a monotone real circuit of size s using k-ary monotone gates then it can be computed by a monotone real circuit (with real gates with fan-in 2) of size $O(sn^{k-2})$.

Real communication protocols. A beautiful characterization by Karchmer and Wigderson [33] shows that a Boolean function f has a monotone formula of size s if and only if the monotone Krachmer-Wigderson (KW) game associate with f (see Definition 3.1) has communication complexity $\log s$. Krajíček [36] defined real communication protocols in which the 2 parties have access to a greater-than oracle, and proved that the real communication complexity of the monotone KW game of a function f is at most logarithmic in the size of the monotone real formula that computes f. Hrubeš and Pudlák (HP) [29] considered a restricted class of real communication protocols and showed that, for every monotone function f, the minimal real communication complexity of monotone KW game that can be achieved by such protocols equals to the monotone real circuit complexity of f. (It is unknown whether any Krajíček's type protocol can be translated into an HP-type protocol.) Chattopadhyay et al. [16] proved a lower bound of $\Omega(n)$ on the complexity of a real communication protocol for an n-bit function; however their lower bound is not for the monotone KW game of a function and therefore it does not imply lower bounds for monotone real formulas.

Balancing formulas. There are many papers showing how to balance a formula starting with the work of Spira [50], who proved that any Boolean formula F of size s can be simulated by an equivalent formula of depth $O(\log s)$. There are several results improving or extending Spira's theorem, e.g., [13, 47, 56, 14, 53, 15, 23]. Specifically, Wegener [56] proved the statement for monotone Boolean formulas. Furthermore, Gál and Jang [23] showed how to balance circuits with small segregators, and, in particular, circuits with small separators.

Lower bounds for secret-sharing schemes. The best known lower bound on the share size of secret-sharing schemes is far from the exponential upper bounds on the share size described above. Csirmaz [18, 19] proved that there is an explicit monotone function $f : \{0,1\}^n \to \{0,1\}$ that requires total share size of $\Omega(n^2/\log n)$ times the size of the secret in any secret-sharing scheme realizing it. No better lower bounds are known for secret-sharing schemes (even for non-explicit monotone functions). Better lower bounds are known for *linear* secret-sharing schemes, which are schemes based on monotone span programs [34]. Pitassi and Robere [42] showed an explicit n-bit function (for every n) that requires share size of $2^{\Omega(n)}$ times the length of the secret in any *linear* secret-sharing scheme realizing it. Furthermore, Babai, Gál, and Wigderson [5] showed that for almost all monotone functions the share size in any *linear* scheme for one-bit secrets over any finite field is $\Omega(2^{n/2})$ times the length of the secret. Furthermore, Beimel and Ishai [11] observed that if a monotone function can be realized by an efficient linear secret-sharing scheme, then the function has a (non-monotone) NC-circuit.

2 Preliminaries

In this section we define the basic notions used in this work.

Monotone Boolean functions. Let $a, b \in \{0,1\}^n$ be a pair of equal-length strings. We say that $a \leq b$ if $a_i \leq b_i$ for every $i \in [n]$, and say that $a < b$ if, in addition, $a_j < b_j$ for some $j \in [n]$. A function $f : \{0,1\}^n \to \{0,1\}$ is monotone if for every $a \leq b \in \{0,1\}^n$, it holds that $f(a) \leq f(b)$. A minterm of a monotone function f is an assignment b such that $f(b) = 1$ and $f(a) = 1$ for every $a < b$. Similarly, a maxterm of $f : \{0,1\}^n \to \{0,1\}$ is an assignment b such that $f(b) = 0$ and $f(a) = 1$ for every $a > b$. A monotone function is fully defined by its minterms (resp., maxterms).

We move on and define monotone real formulas and circuits as introduced in [43].

Definition 2.1 (Monotone real circuits and formulas)**.** *A* monotone real function *$f : \mathbb{R}^n \to \mathbb{R}$ is a real function in which for every two inputs $x = (x_1, \ldots x_n), x' = (x'_1, \ldots, x'_n) \in \mathbb{R}^n$ such that $x_i \leq x'_i$ for every $i \in [n]$, it holds that $f(x) \leq f(x')$. A* monotone real gate *G takes as an input n values $x_1, \ldots, x_n \in \mathbb{R}$, computes some monotone real function $f : \mathbb{R}^n \to \mathbb{R}$, and returns $f(x_1, \ldots, x_n)$ as an output. A* monotone real circuit (MRC) *C is a circuit in which each gate is a monotone real gate G with fan-in 2 and for every input $x \in \{0,1\}^n$ the output of the circuit C is Boolean. A* monotone real formula (MRF) *is a monotone real circuit whose DAG is a tree.*

Note that in an MRC/MRF the inputs and outputs are Boolean, while the values on internal edges can be any real numbers. We allow AND and OR gates and other Boolean gates in an MRC with the convention that their inputs are always Boolean. Taking monotone real gates with fan-in 2 is the more common definition of MRCs and it will help us prove our lower bounds. Furthermore, in our constructions of MRFs the fan-in of all gates is 2.

We continue with the definition of slice gates and formulas over slice gates. Throughout the paper, we denote the Hamming weight of a string y by $\mathrm{wt}(y)$.

Definition 2.2 (Slice gates and formulas over slice gates)**.** *A (k, n)-slice function $f : \{0,1\}^n \to \{0,1\}$ is a monotone function such that for every $y \in \{0,1\}^n$:*

- *If $\mathrm{wt}(y) < k$, then $f(y) = 0$.*
- *If $\mathrm{wt}(y) = k$, then $f(y)$ can be either 0 or 1.*
- *If $\mathrm{wt}(y) > k$, then $f(y) = 1$.*

We refer to k as the weight of the gate. A (k, n)-slice gate is a monotone gate computing a (k, n)-slice function. A formula over slice gates (FOS) is a formula F whose gates are slice gates; we stress that each slice gate in F can have different values for k and n (and in particular the fan-in of each slice gate is arbitrary).

Example 2.3. An AND gate with n inputs is an (n, n)-slice gate.[5] An OR gate with n inputs is a $(1, n)$-slice gate. Another example of a slice gate computing a k-threshold function (i.e., computing the function $\mathrm{TR}_k : \{0,1\}^n \to \{0,1\}$ such that $\mathrm{TR}_k(y) = 1$ if and only if the string y contains at least k ones). However, slice gates can compute a richer class of functions and the number of (k, n)-slice functions is $2^{\binom{n}{k}}$.

In this paper, we define the size of a circuit/formula as the number of *gates* in the circuit/formula (including input gates). This convention is used both for circuits with monotone real gates and for formulas over slice gates. We note that since monotone real circuits have fan-in 2, our definition of monotone real circuit size is essentially equivalent to the definition that counts the total number of edges in the circuit. Furthermore, the same is true for formulas.

We recall the definition of generalized secret-sharing schemes.

Definition 2.4. *An n-party* secret-sharing scheme*, with domain of secrets S such that $\{0,1\} \subseteq S$ and finite domains of shares $S_1, \dots, S_n$, is a randomized (possibly inefficient) algorithm $\mathcal{D}$ that maps a secret $s \in S$ to a vector of shares $(s_1, \dots, s_n) \in S_1 \times \cdots \times S_n$. We say that $\mathcal{D}$ realizes a (possibly partial) monotone function f over $\{0,1\}^n$ if for every $x \in \{0,1\}^n$ and every pair of secrets $s, s' \in S$ the random variables $(s_1, \dots, s_n)$ obtained by invoking $\mathcal{D}$ on s, and the random variables $(s'_1, \dots, s'_n)$ obtained by invoking $\mathcal{D}$ on s' satisfy the following properties:*

Correctness. *If $f(x) = 1$ then the random variables $s_x = (s_i)_{i:x_i=1}$ and $s'_x = (s'_i)_{i:x_i=1}$ have disjoint supports, that is one can recover the secret from the shares s_x.*

Privacy. *If $f(x) = 0$ then the random variable s_x is identically distributed to the random variable s'_x, that is, the shares s_x do not disclose any information on the secret.*

The secret size *in a secret-sharing scheme $\mathcal{D}$ is defined as* $\log |S|$*, the* share size *of the scheme $\mathcal{D}$ is defined as the size of the largest share, i.e.,* $\max_{1 \le i \le n}\{\log |S_i|\}$*, and the total share size is defined as the sum of the share sizes , i.e.,* $\sum_{1 \le i \le n} \log |S_i|$*. The* maximal information ratio *(resp.,* total information ratio*) of the scheme is defined as the ratio between the maximal share size (resp., total share size) and the secret size.*[6]

For more information on secret-sharing schemes, one can refer to, e.g., [6].

[5]It is also an $(n-1, n)$-slice gate.

[6]The maximal/total share-size measures essentially ignore the bit-length of the secret, whereas the maximal/total information-ratio measures normalize the bit length of the longest share/sum of the shares by the length of the secret, and are therefore more suitable to the case of long secrets.

3 Lower Bounds for Formula Size over Slice Gates

In this section, we prove Theorem 1.2 by showing that there exists a function in monotone NP that requires size $2^{\Omega(n/\log^2 n)}$ formulas over slice gates.

Our result goes through monotone real circuits. First, a result of Rosenbloom [46] shows that any slice function over k bits can be computed by an $O(k)$-size $O(\log k)$-depth read-twice monotone real formulas. Therefore, a formula over slices of size s can be transformed into a monotone real *circuit* of size $O(s)$. While this transformation preserves the size it may blow up the *depth* of the circuit.[7] This is unfortunate since we only know how to prove strong (almost-exponential) lower-bounds against *low-depth* circuits.

To overcome this problem, we observe that the monotone real circuit that we get can be separated into smaller sub-circuits by deleting 2 gates. This fact enables us to construct a balanced real protocol for the monotone Karchmer-Wigderson (KW) game whose complexity is $O(\log s)$. (See Section 3.1.) We then prove a lower bound on the complexity of real protocols for the monotone KW game of an explicit function, using a lower bound of Göös and Pitassi [26] on the randomized monotone KW game of this function. (See Section 3.2.) By combining these steps, we obtain $2^{\Omega(n/\log^2 n)}$ size lower bounds on the size of separable monotone real circuits for an explicit function, thus, implying the same lower bounds for formulas over slices.

3.1 Converting a Formula over Slice Gates to a Real Protocol for the Monotone KW Game

To prove our results, we need the following definitions.

Definition 3.1 (Monotone KW games [33]). *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function. The monotone KW game associated with f is a two-player communication game. Alice receives an input $u \in f^{-1}(1)$ and Bob receives an input $v \in f^{-1}(0)$, and they communicate in order to find an index i such that $u_i > v_i$.*

Definition 3.2 (Real communication protocols). *In a real communication protocol, deterministic Alice and Bob interact via a referee. At the start Alice has a binary string $u \in U \subseteq \{0,1\}^n$ and Bob has $v \in V \subseteq \{0,1\}^n$. At round i, Alice and Bob each send a real number $a_i(u)$ and $b_i(v)$, respectively, to a referee, where $a_i(u)$ depends on u and the bits sent by the referee so far, and similarly $b_i(v)$ depends on v and the bits sent by the referee so far. The referee sends to both players 1 if $a_i(u) > b_i(v)$ and otherwise it sends 0. Each player does not see the numbers sent by the other player. At the end of the protocol both players should know the value of the function or the same solution to the search problem that they are solving. The complexity of the protocol is the maximum number of rounds (or equivalently the number of bits sent by the referee) over all inputs of Alice and Bob of length n.*

Krajíček [36] showed that if a function has a monotone real formula of size s, then the associated monotone KW game can be solved by a real protocol with complexity $O(\log s)$. We generalize this result to monotone real circuits that have small separators. A similar result for Boolean circuits has been proved by Gal and Jang [23]. In the following, we say that a directed-acyclic graph (DAG) $G = (V, E)$ has a root (or a source) if there exists a vertex $s \in V$ such that for every $v \in V$ there is a path from s to v in G. Clearly, a DAG has at most one root. We say that a vertex v is reachable from a vertex u if there is a path from u to v.

[7] To illustrate this point, consider a balanced formula over slices of size $s = O(2^{n^{0.8}})$ that consists of slice gates whose fan-in is $2^{n^{0.5}}$ that are connected sequentially in a path of length $n^{0.8}$. (All other gates are fan-in 2 gates.) Each slice gate can be replaced by Rosenbloom's monotone real read-twice formula whose depth is $O(n^{0.5})$, leading to a monotone real circuit of depth $O(n^{0.5} \cdot n^{0.8}) > n$.

Definition 3.3. *A DAG $G = (V, E)$ with a root is (α, k)-separable if for every sub-graph $G' = (V', E')$ of G (i.e., $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$) that has a root there exist k vertices $a_1, \ldots, a_k$ in V' such that:*

- *For every $\ell \in [k]$, the number of vertices reachable in G' from a_ℓ is at most $\alpha|V'|$.*
- *If we remove the out-going edges of the vertices $a_1, \ldots, a_k$ from G', then the number of vertices reachable from the root of G' in the resulting graph is at most $\alpha|V'|$.*

Example 3.4. A well-known result states that every directed binary tree is $(2/3, 1)$-separable, i.e., it contains a vertex that separates the tree to two components, each component of size at most $2/3$ of the size of the original tree. To see this, we start at the root of the tree T and follow a path through the tree, always going to the sub-tree of larger size. The procedure stops whenever we hit a vertex u such that the sub-tree, T_u, rooted at u has a size less than $2/3$ times the size of the entire tree T. Since u is the largest child of a vertex whose sub-tree has size at least $2/3$ times the size of T, it follows that T_u has size at least $1/3$ times the size of T, and therefore we can separate T into two components, $T - T_u$ and T_u, where each component has size between $1/3$ and $2/3$ times the size of T.

We next prove that for every monotone real circuit of size s that is separable, the monotone KW game of the function computed by the circuit has a real protocol with complexity $O(\log s)$. Specifically, we use the balancing technique, introduced by Spira [50] for Boolean formulas and used by Krajíček [36] for constructing real protocols from monotone real formulas.

Lemma 3.5. *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function computed by a monotone real circuit C of size s. If the DAG of C is (α, k)-separable, then the monotone KW game associated with f can be computed by a real protocol with complexity $O(k \log_{1/\alpha} s)$.*

Proof. We use C to construct a real protocol for the monotone KW game with complexity $O(k \log_{1/\alpha} s)$. In this protocol, Alice is given $u \in f^{-1}(1)$, Bob is given $v \in f^{-1}(0)$, and they want to find an index j such that $u_j > v_j$.

We first make the following easy but important observation. The function h computed at the root of C has the property that $h(u) > h(v)$. Furthermore, for every internal vertex a of C with children b, c, if $h_a(u) > h_a(v)$ (where h_a is the function computed at vertex a), then either $h_b(u) > h_b(v)$ or $h_c(u) > h_c(v)$. This holds by monotonicity, because $h_a(x)$ is by definition a monotone function of $h_b(x)$ and $h_c(x)$.

For a circuit C, we consider the rooted DAG G whose vertices are the gates of the circuit (including the input gates) and for each internal gate there are edges directed from the gate to its input gates. Given inputs u, v, we color each vertex a of G by Red or Blue, where Red means that the function computed at this vertex has $h_a(u) > h_a(v)$ and Blue otherwise. We say that a path is Red if all its vertices are Red. By the above observation, the root of G is colored Red, and for each vertex that is Red, at least one of its children is Red, and thus there must exist a Red path from any Red node (in particular, the root) to a Red leaf. A Red leaf is what we are after since each leaf is labeled by a coordinate j and if it is Red, then we must have $u_j > v_j$ as desired. This leads to a simple real protocol where Alice and Bob traverse a Red path from the root to a leaf; however, the complexity of this protocol is the depth of G (i.e., the maximal length of a path from the root to a leaf), which can be $O(s)$.

We design an efficient real protocol finding a Red leaf in iterations, using the fact that the players can determine in one round whether any particular vertex is colored Red or Blue. At iteration i, the parties hold a sub-graph G_i of G of size at most $\alpha^i s$; this sub-graph has a root whose color is Red and contains a Red path from every red vertex to a Red leaf. So after $O(\log_{1/\alpha} s)$ iterations, Alice and Bob will have arrived at a Red leaf labeled by some coordinate j where $u_j > v_j$ as desired.

Iteration $i = 0$. Alice and Bob are at the root vertex of the graph G that computes f and by definition it is Red.

Iteration i. In the beginning of the iteration, Alice and Bob are at a Red vertex rooted at a sub-graph G_i of G of size at most $\alpha^i s$ and do the following:

1. Find k vertices $a_1, \dots, a_k$ that separate the sub-graph G_i.
2. For $\ell = 1$ to k do:
   - Alice locally computes the value of the monotone function computed by vertex a_ℓ in C on her input u, and similarly Bob locally computes the value on his input v. They send these values to the referee, who tells them which is larger. If Alice's value is larger, then a_ℓ is Red, so they take $G_{i+1} = G_{a_\ell}$, the sub-graph rooted at a_ℓ in G_i, and continue to the next iteration.
3. Otherwise, $a_1, \dots, a_\ell$ are Blue. Alice and Bob take the sub-graph G_{i+1}, obtained from G_i by removing all out-going edges of a_j for each $j \in [\ell]$, and removing all vertices not reachable from the root of G_i. Clearly, the G_{i+1} is rooted DAG whose root is Red. As we removed sub-graphs rooted at Blue vertices, each Red vertex in G_{i+1} has a Red path to a leaf.

In each of the cases in the iteration i, the number of vertices of the sub-graph G_{i+1} is at most α times the number of vertices G_i. Thus after $O(\log_{1/\alpha} s)$ iterations, Alice and Bob reach a Red leaf. As each iteration contains at most k rounds, the theorem follows. □

We show that if f has a monotone formula over slice gates of size s, then the real communication complexity of the associated monotone KW game is at most $O(\log s)$. By Lemma 3.5, it suffices to show that every monotone formula over slice gates of size s can be converted to a monotone real circuit of size $O(s)$ whose DAG is $(5/6, 2)$-separable. This is done using the following result of Rosenbloom [46], showing that monotone real formulas can compute the class of all slice functions very efficiently. We provide a proof sketch of this result since we use specific properties of Rosenbloom's construction.

Theorem 3.6 ([46]). *Every slice gate with fan-in t can be computed by a read-twice fan-in-2 monotone real formula of size $O(t)$ and depth $O(\log t)$.*

Proof sketch. Given $x = (x_1, \dots, x_t)$, associate with it two integers $p(x) = \mathrm{wt}(x) \cdot 2^t + b(x)$ and $m(x) = \mathrm{wt}(x) \cdot 2^t - b(x)$ where $\mathrm{wt}(x)$ is the number of 1's in x and $b(x)$ is the integer represented by the string x, i.e., $b(x) = \sum_{i=1}^t 2^{i-1} x_i$. The mapping $x \mapsto (p(x), m(x))$ has the following useful feature. For every pair of distinct strings $u \neq v$, if $\mathrm{wt}(u) < \mathrm{wt}(v)$, then the pair $(p(u), m(u))$ is strictly smaller than the pair $(p(v), m(v))$ (i.e., both $p(u) < p(v)$ and $m(u) < m(v)$); On the other hand, if $\mathrm{wt}(u) = \mathrm{wt}(v)$, then the pair $(p(u), m(u))$ is incomparable to the pair $(p(v), m(v))$ (i.e., $p(u) < p(v)$ if and only if $m(u) > m(v)$).

Now if f is a slice function (defined on inputs of weight k) then there is a monotone function G from $\mathbb{R}^2$ to $\{0,1\}$ such that $G(p(x), m(x)) = f(x)$ for all $x \in \{0,1\}^n$ such that $\mathrm{wt}(x) = k$. Furthermore, $p(x) = \sum_{i=1}^t (2^t + 2^{i-1}) x_i$ and $m(x) = \sum_{i=1}^t (2^t - 2^{i-1}) x_i$. Thus, both $p(x)$ and $m(x)$ can be computed by a binary tree whose vertices compute (weighted) addition over the reals. Thus any slice-gate can be simulated by a monotone real formula with addition gates computing $p(x)$ and $m(x)$ and the top real gate computing G on these inputs. □

Let $m(x)$ and $p(x)$ be the functions from the proof sketch of Theorem 3.6. In the sequence, we will refer to the tree computing $m(x)$ as the left tree and to the tree computing $p(x)$ as the right tree. Furthermore, for each vertex a in the left tree, we will refer to the analogous vertex in the right tree as the twin of a.

Given a formula over slice gates, we can replace each slice gate with the monotone real formula of Rosenbloom. However, since this formula is read-twice, we get a monotone real *circuit*. Thus, we cannot directly apply the results of [36] that hold for monotone real formulas to obtain a lower bound for formulas over slice gates. We exploit the structure of the circuit and the structure of Rosenbloom's formula to prove that the DAG of the resulting monotone real circuit is separable by two vertices.

Lemma 3.7. *Let $f : \{0,1\}^n \to \{0,1\}$ be a monotone function computed by a size s formula over slice gates and monotone real gates. Then the monotone KW game associated with f can be computed by a real protocol with complexity $O(\log s)$.*

Proof. Let F be a size s formula over slice gates and monotone real gates computing f. We replace each slice gate in F with the read-twice monotone real formula of Rosenbloom and get a monotone real *circuit* C of size $O(s)$ computing f. We next prove that $G = (V, E)$ – the DAG of C – is $(5/6, 2)$-separable. Let $G' = (V', E')$ be a sub-graph of G that contains a root. Construct from G' a tree $T = (V_T, E_T)$ by merging each pair of twins in G' to one vertex (if a vertex does not have a twin in G' we keep it in the tree). Clearly, $0.5|V'| \leq |V_T| \leq |V'|$. As in Example 3.4, T has a vertex a that separates it to two sub-trees of size at least $1/3|V_T|$ and at most $2/3|V_T|$. If a is a merge of two twins a_1, a_2 in G', then take these two twins as the separating set in G'. Otherwise take a as the separating set. See Figure 1 for an illustration of such a graph and a separating set in it. As the number of vertices reachable in T from a is at most $2/3|V_T|$, the number of vertices *not reachable* from the root of T after removing a is at least $1/3|V_T| \geq 1/6|V'|$. Thus, the number of vertices *not reachable* from the root of G' after removing the separating set is at least $1/6|V'|$. Similarly, the number of vertices not reachable by the vertices in the separating set is at least $1/3|V_T| \geq 1/6|V'|$. This implies that the number of vertices reachable in G' by the vertices in the separating set and the root is at most $5/6|V'|$.

Since the DAG of C is $(5/6, 2)$-separable, then by Lemma 3.5, the monotone KW game associated with f can be computed by a real protocol with complexity $O(\log s)$. □

3.2 Completing the Proof of the Lower Bounds for Formula Size over Slice Gates

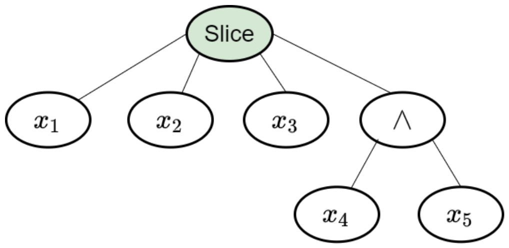
Next we show that real protocols can be simulated by randomized protocols in the plain model.[8] This lemma was originally proved in [36].

Lemma 3.8. *A real communication protocol for the monotone KW game for $f : \{0,1\}^n \to \{0,1\}$ with complexity d implies a randomized communication protocol for the monotone KW game with complexity $O(d \log n)$.*

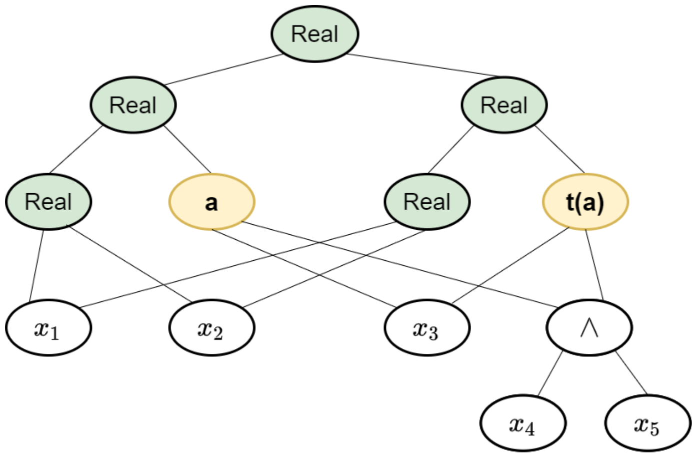
Proof. If $d \geq n/\log n$ the theorem is trivial since the monotone KW game can be solved by a (deterministic) protocol with complexity $O(n)$. Thus, in the sequence we assume that $d \leq n/\log n$.

We will show that every round of a real protocol can be simulated by a randomized communication protocol of cost $O(\log n)$. Recall that a round in the real protocol consists of Alice and Bob each sending arbitrary real numbers a, b (which depend on their respective inputs and the communication so far) to a referee, who responds with 1 if $a > b$ and 0 otherwise. Although these values can be any real numbers, in each round i there are at most 2^n values $c_1, c_2, \ldots, c_{2^n}$ that Alice and Bob can send to the referee in

[8] For the definition of randomized protocols see, e.g., [37].



An example of a simple slice formula F. The formula has 5 input bits and one slice gate with fan-in 4.



The formula F after the Rosenbloom transformation is applied to its slice gate. The slice gate becomes a tree of real gates, and the DAG structure is transformed from a formula to a circuit. The (real) gates a and its twin $t(a)$ are an example of separators for the circuit's DAG.

Figure 1: An example of a balancing step that goes through the Rosenbloom transformation.

the (deterministic) real protocol (i.e., one value per each input). Assume that these values are sorted, i.e., $c_1 < c_2 < \cdots < c_{2^n}$. Assume that in round i, Alice sends c_j and Bob sends c_k, and the referee returns 1 if $c_j > c_k$, which is true if and only if $j > k$. Thus, we can replace the message of Alice by j and the message of Bob by k, i.e., all numbers are n bit strings. Since this is just the greater-than (GT) function, it can be computed by a randomized protocol for GT, whose complexity for an error ε is $O(\log n + \log \varepsilon^{-1})$ [41, 54]. We will want the overall error to be bounded by a constant, so we will set $\varepsilon = O(1/d)$. Thus, the GT protocol for simulating one round costs $O(\log n + \log d) = O(\log n)$ (since $d \leq n/\log n$) and the cost of simulating d rounds is $O(d \log n)$. □

Göös and Pitassi [26] proved that there is a function f in monotone NP that requires monotone circuit depth $\Omega(n/\log n)$, and therefore monotone formula size $2^{\Omega(n/\log n)}$. This is equivalent to proving that the deterministic communication complexity of the monotone KW game of f is $\Omega(n/\log n)$. However, Göös and Pitassi proved that the lower bound also applies to the randomized communication complexity of the monotone KW game of f; this gives the best known lower bound for the randomized complexity of a monotone KW game of a function.

Theorem 3.9 (Implicit in [26])**.** *There is a function f in monotone NP such that the randomized communication complexity of the monotone KW game for it has complexity $\Omega(n/\log n)$.*

We are ready to complete the proof of Theorem 1.2, the lower bound on the size of a formula over slice gates and monotone real gates.

Proof of Theorem 1.2. Consider the function f from Theorem 3.9, and suppose for contradiction that f is computable by a formula over slice gates and monotone real gates of size $2^{o(n/\log^2 n)}$. By Lemma 3.7, this implies that the monotone KW game for f has a real communication protocol of cost $o(n/\log^2 n)$, and by Lemma 3.8, the monotone KW game for f has a constant-error randomized protocol of cost $o(n/\log n)$. But this contradicts Theorem 3.9, and thus f requires size $2^{\Omega(n/\log^2 n)}$ formulas over slice gates and monotone real gates. □

4 Secret Sharing and Monotone Real Computation are Incomparable

In this section, we show that for some monotone functions f, there are provable gaps between the secret-sharing complexity (measured as the total share size of a secret sharing scheme that realizes f), the MRC complexity, and the MRF complexity. Thus, we separate these complexity measures.

4.1 Secret Sharing can be Super-Polynomially Cheaper than Monotone Real Circuits

Let OddFactor_n denote the monotone function that takes $n = v^2$ inputs representing the edges of a bipartite graph X with v vertices in each part, and outputs 1 if and only if the graph X has an odd factor, i.e., a spanning sub-graph such that all vertices have an odd degree in the sub-graph. Existing results can be used to show that the function OddFactor_n demonstrates a super-polynomial gap between secret-sharing complexity and Monotone Real Circuits complexity.

Theorem 4.1. *The function $\mathit{OddFactor}_n$ has a secret-sharing scheme with total share size n, but any MRC that computes $\mathit{OddFactor}_n$ must be of size $n^{\Omega(\log n)}$ and any MRF that computes $\mathit{OddFactor}_n$ must be of size $2^{\Omega(\sqrt{n}/\log n)}$. Moreover, the latter bound holds also for formulas that employ both real gates and slice gates.*

Proof. By [5], $\mathrm{OddFactor}_n$ can be realized by a linear secret-sharing scheme with a one-bit secret, a one-bit share per party, and total share size n. In the same paper it is shown that $\mathrm{OddFactor}_n$ requires a monotone circuit of size $n^{\Omega(\log n)}$, by reducing it to a lower bound by Razborov [44] for the perfect matching function. Fu [22] extended Razborov's lower bound to monotone real circuits.

To prove the last part, we note that [5] also show that $\mathrm{OddFactor}_n$ requires a monotone *formula* of size $2^{\Omega(\sqrt{n})}$. This lower bound goes through a lower bound of $\Omega(\sqrt{n})$ for the deterministic monotone KW game of $\mathrm{OddFactor}_n$. The proof of [5] is by reduction to the randomized communication complexity of the disjointness function, and actually extends to randomized monotone KW games. Therefore, by Lemmas 3.7 and 3.8, we can get a lower bound of $2^{\Omega(\sqrt{n}/\log n)}$ for the size of monotone real formulas (that may also employ slice gates) for $\mathrm{OddFactor}_n$. □

4.2 Monotone Real Formulas can be Cheaper than Secret Sharing

We prove the following theorem.

Theorem 4.2. *There exists a monotone function that can be computed by an MRF of size $O(n)$ but requires a total share size of $\Omega(n^2/\log n)$ for any secret-sharing scheme. Moreover, the function has $O(n^2)$ min-terms and therefore it can be computed by a polynomial-size monotone DNF.*

Improving the gap in Theorem 4.2 requires proving better lower-bounds of $\omega(n^2/\log n)$ for secret sharing schemes – a task that remains open since Csirmaz's works in the mid-nineties [18, 19]. It should be mentioned, however, that there are candidates that seem to demonstrate a gap of $2^{\tilde{\Omega}(\sqrt{n})}$ (e.g., slice functions) or even a gap of $2^{\Omega(n)}$ (see Appendix A).

The following function will be used as a building block in the gap theorem.

Definition 4.3 (The simple Csirmaz function C_n [18])**.** *For every $n \in \mathbb{N}$, let k be the largest integer such that $2^k \leq n$. The function C_n is parameterized by some non-increasing ordering $(y^0, \ldots, y^{2^k-1})$ of all strings of length k. Here non-increasing means that*

$$\text{for every } i < i', \quad \text{it holds that } y^i \not\leq y^{i'}. \tag{1}$$

The function $C_n : \{0,1\}^{n+k} \to \{0,1\}$ is the monotone function whose min-terms are $1^i \circ 0^{n-i} \circ y^i$ for $i = 0, \ldots, 2^k - 1$, that is, the i-th minterm contains i ones concatenated with $n - i$ zeros, concatenated by y^i.

The simple Csirmaz function is not fully defined as the order of the strings $(y^0, y^1, \ldots, y^{2^k-1})$ is not specified. In the next claim we choose a specific order that will enable us to construct a small MRF for it. The construction borrows ideas from [46] (see Theorem 3.6).

Claim 4.4. *There exists a non-increasing ordering over k-bit strings for which the corresponding function C_n has an MRF F of size $O(n)$. Moreover, if we parse the input to the function as $(x, y) \in \{0,1\}^n \times \{0,1\}^k$, the MRF will have the following form:*

$$F(x, y) := G(F_b(x), F_p(y)),$$

where the size of F_b is $O(n)$, the size of F_p is $O(k)$, and G is a monotone real gate.

Proof. We define the following ordering of the strings of length k using the function p defined above:

$$p(y_1, \ldots, y_k) = \sum_{i=1}^{k} (2^k + 2^{i-1}) y_i,$$

that is $p(y_1, \ldots, y_k) = \mathrm{wt}(y_1, \ldots, y_k) \cdot 2^k + b(y_1, \ldots, y_k)$, where $\mathrm{wt}(y_1, \ldots, y_k)$ is the weight of a string, and $b(y_1, \ldots, y_k)$ is the integer represented by the string $(y_k, \ldots, y_1)$. We order the strings according to their p-value from largest to smallest (i.e., $y^1 = 1^k$ is the k-bit string that achieves the maximal value of p among all k-bit strings). This order is well defined since p is injective. We next argue that if $i < i'$ then $y^i \not< y^{i'}$ as required by the definition. We prove the counter-positive. If $y^i < y^{i'}$, then $\mathrm{wt}(y^i) < \mathrm{wt}(y^{i'})$, which implies that $p(y^i) < p(y^{i'})$ since each 1 in the input contributes to p a huge summand of 2^k and $b(y^i) < 2^k$. It follows that $i' < i$, as required.

Before constructing an MRF for C_n we make the following observation. Parse the input to C_n as $(x, y) \in \{0,1\}^n \times \{0,1\}^k$. Recall that $C_n(x \circ y) = 1$ if and only if $x \circ y$ is bigger than some minterm $1^j \circ 0^{n-j} \circ y^j$ of C_n. Letting i denote the index for which $y = y^i$, this happens if and only if $x \geq 1^j \circ 0^{n-j}$ and $y^i \geq y^j$, thus $j \geq i$. Thus, $C_n(x \circ y^i) = 1$ if and only if the first i bits of x are 1. We will use this characterization in order to compute C_n.

Let F_b and F_p be MRFs that compute the functions $b : \{0,1\}^n \to \mathbb{R}$ and $p : \{0,1\}^k \to \mathbb{R}$ respectively. Recall that b returns the integer represented by $x = (x_1, \ldots, x_n)$ with the first bits being the most significant ones and notice that the first i bits in $x = (x_1, \ldots, x_n)$ are 1 if and only if $b(x_1, \ldots, x_n) \geq \sum_{j=n-i}^{n} 2^{j-1}$. Further, observe that both b and p can be realized with complexity of $O(n)$ and $O(k)$, respectively. (In both cases, we simply use a tree over-weighted addition gates.) Consider the formula

$$F(x, y) := G(F_b(x), F_p(y)),$$

where $G(u, v)$ is a real gate that acts as follows: For $v \in \{0, \ldots, 2^{k-1}\}$, recover i, s.t., $v = p(y^i)$ and then output 1 if $u \geq \sum_{j=n-i}^{n} 2^{j-1}$ and output 0 otherwise. By the above observations, F computes C_n.

It remains to show that $G(u, v)$ is a monotone function.[9] Clearly, $G(u, v)$ is monotone in u. To see that G is monotone in its second argument, assume $v > v'$ and there is some u such that $G(u, v') = 1$. We need to prove that $G(u, v) = 1$. Let $v = p(y^i)$ and $v' = p(y^{i'})$. Since $p(y^i) = v > v' = p(y^{i'})$ and $G(u, v') = 1$, it must be that $i < i'$ and $u \geq \sum_{j=n-i'}^{n} 2^{j-1} > \sum_{j=n-i}^{n} 2^{j-1}$, thus $G(u, v) = 1$. Overall, F is an MRF of size $O(n + k) = O(n)$. □

Csirmaz [18] proved that in any secret-sharing scheme realizing the function C_n there is at least one party whose share size is $\Omega(n/\log n)$. (This lower bound holds for any order satisfying (1).) Based on C_n, Csirmaz later introduced in [19] the following function, C'_n, and showed that in any secret-sharing scheme realizing this function the total share size is $\Omega(n^2/\log n)$.

Definition 4.5 (The full Csirmaz function C'_n)**.** *For every $n \in \mathbb{N}$, define a monotone function C'_n over inputs in $\{0,1\}^{2n}$ as follows: Let k be the largest integer such that $2^k \leq n$ and $L = \lfloor n/k \rfloor$, and define*

$$C'_n(x_1, \ldots, x_{2n-k\cdot L}, y_{1,1}, \ldots, y_{1,k}, \ldots, y_{L,1}, \ldots, y_{L,k}) = \bigvee_{\ell=1}^{L} C_n(x_1, \ldots, x_n, y_{\ell,1}, \ldots, y_{\ell,k}).$$

[9]Formally speaking, we only defined G over the domain $\mathbb{R} \times [0, 2^{k-1}]$ and we will show that it is monotone over this domain. One can then easily extend G to $\mathbb{R} \times \mathbb{R}$ while maintaining monotonicity.

Lemma 4.6 (MRF for the full Csirmaz function)**.** *There exists a non-increasing ordering over k-bit strings for which the corresponding function C'_n has an MRF of size $O(n)$.*

Proof. Define the following MRF:

$$F'(x_1,\ldots,x_{2n-k\cdot L},y_{1,1},\ldots,y_{1,k},\ldots,y_{L,1},\ldots,y_{L,k}) = G\left(F_b(x_1,\ldots,x_n), \max_{1\le\ell\le L}\{F_p(y_{\ell,1},\ldots,y_{\ell,k})\}\right),$$

where $F_b(x)$, $F_p(y)$, and $G(\cdot,\cdot)$ are the MRFs that were defined in Claim 4.4. We claim that F' computes C'_n. If $C'_n(x_1,\ldots,x_{2n-k\cdot L},y_{1,1},\ldots,y_{1,k},\ldots,y_{L,1},\ldots,y_{L,k}) = 1$, then there exists an ℓ_0 such that $C_n(x_1,\ldots,x_n,y_{\ell_0,1},\ldots,y_{\ell_0,k}) = 1$, thus, by Claim 4.4, $G(F_b(x_1,\ldots,x_n),F_p(y_{\ell_0,1},\ldots,y_{\ell_0,k})) = 1$, and (since G is monotone) F' returns 1. On the other hand, if $F'(x_1,\ldots,x_{2n-k\cdot L},y_{1,1},\ldots,y_{1,k},\ldots,y_{L,1},\ldots,y_{L,k}) = 1$, then let ℓ_0 be such that $F_p(y_{\ell_0,1},\ldots,y_{\ell_0,k}) = \max_{1\le\ell\le L}\{F_p(y_{\ell,1},\ldots,y_{\ell,k})\}$, thus, $G(F_b(x_1,\ldots,x_n),F_p(y_{\ell_0,1},\ldots,y_{\ell_0,k})) = 1$; by Claim 4.4, $C_n(x_1,\ldots,x_n,y_{\ell_0,1},\ldots,y_{\ell_0,k}) = 1$, i.e., C'_n returns 1.

Recalling that the size of F_b and F_p is linear in the number of corresponding inputs, we conclude that the total complexity of F' is $1 + |F_b| + L\cdot|F_p| = O(n + L\cdot k) = O(n)$ (as $L = \lfloor n/k\rfloor$). □

As already mentioned, by [19], the total share size in any secret-sharing scheme realizing C'_n is $\Omega(n^2/\log n)$. Furthermore, it is not hard to verify that has at most $O(n^2)$ min-terms (since C_n has only $O(n)$ min-terms), and so C'_n can be computed by a polynomial-size monotone DNF. Thus, Theorem 4.2 follows from Lemma 4.6.

5 Recent Secret-Sharing Schemes Imply Slice Formulas

In this section we argue that all the known non-linear constructions [38, 2, 3, 4, 8] of secret-sharing schemes with non-trivial share size 2^{cn}, for a constant $c < 1$, realize a monotone function f via a constant-depth monotone formula F of size 2^{cn} over the standard (unbounded fan-in) AND/OR basis enriched with a special type of *partite* (k,N)-slice gates whose N-bit input is partitioned into k equal-sized blocks of bit length $B = N/k$ and where we only "care" about the value of the gate when it is applied to an input that has exactly one single bit "turned-on" in each block. (See Definition 5.1.) These (k,N)-slice gates will typically be employed with $k = n^{\varepsilon}$ and fan-in $N = n^{\varepsilon}\cdot 2^{n^{1-\varepsilon}}$ for some $\varepsilon < 1$. For this regime, these slice gates can be realized by a secret-sharing scheme with share size of $2^{\tilde{O}(\sqrt{n})}$ [39] via so-called *conditional disclosure of secrets* (CDS) protocols [25]. The aforementioned works then use (sometimes implicitly) a generic transformation from formulas to secret sharing (described in Appendix B) that yields a secret-sharing scheme with share size 2^{cn}.

In the subsequent sections, we explain how to interpret the previous works under this framework. Naturally, we do not provide full details for each of these constructions and only provide a "sketched dictionary" that allows one to translate the statements from the original papers to the language of formulas over slices. The interested reader is referred to the original papers for full details.

5.1 The LV Construction

For a balancing parameter $\delta > 0$, the LV construction decomposes a monotone function f into three monotone functions: (1) The "middle part" $f_{\mathrm{mid},\delta}$ that agrees with f on all inputs whose relative Hamming weight is in $(\frac{1}{2}-\delta, \frac{1}{2}+\delta)$ and assigns zero to inputs of smaller weight and one to inputs of larger weight; and (2) A "bottom part", $f_{\mathrm{bot},\delta}$, whose min-terms are all the min-terms of F whose relative weight is at most $\frac{1}{2}-\delta$;

and (3) A "top part", $f_{\text{top},\delta}$, whose maxterms are all the maxterms of F whose relative weight is at least $\frac{1}{2} + \delta$. It is not hard to verify that

$$f = f_{\text{top},\delta} \wedge (f_{\text{mid},\delta} \vee f_{\text{bot},\delta}), \tag{2}$$

and therefore it suffices to realize each of these sub-functions by a small FOS.

The extreme parts, $f_{\text{top},\delta}$ and $f_{\text{bot},\delta}$, can be realized by standard DNF and CNF formulas (respectively) with exponent smaller than 1 since they have at most $\binom{n}{(\frac{1}{2}-\delta)n}$ min-terms and at most $\binom{n}{(\frac{1}{2}+\delta)n}$ maxterms, respectively. Thus, the main effort in [38] is devoted to realizing $f_{\text{mid},\delta}$ with a non-trivial, smaller-than-one, exponent. Towards this end, LV show that the function $f_{\text{mid},\delta}$ can be computed by an exponential-size constant-depth formula with a non-trivial exponent of $\mathbf{M}_{\text{LV}}(\delta) < 1$ that employs standard AND/OR-gates together with a special form of *block-regular* gates. Roughly speaking, in such a gate, $g : \{0,1\}^n \to \{0,1\}$, the n-bit input is partitioned into k equal-sized blocks of size $B = n/k$ each, and the main feature is that g is defined only on inputs $x \in \{0,1\}^n$ that hit exactly b of the indices in each block for some integer parameter b.[10] It will be convenient to specify such a gate g by an arbitrary function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ as follows.

Definition 5.1 (Block regular functions)**.** *Let k, B, b be integers such that $0 < b < B$. We say that a string $x = (x_1, \ldots, x_k) \in (\{0,1\}^B)^k$ is* b-regular *if each block x_i, $i \in [k]$, has Hamming weight of exactly b.*

We say that $g : (\{0,1\}^B)^k \to \{0,1\}$ is a b-regular realization of some specification function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ if the following conditions hold:

- *The function g agrees with $\hat{g}$ on every b-regular input $x = (x_1, \ldots, x_k)$.*
- *The function g rejects all inputs of weight bk that are not b-regular.*

The function g can take arbitrary values on all other inputs.[11] *When $\hat{g}$ is unspecified we refer to g as a b-regular function. A 1-regular function is also referred to as a* partite function *or* (k, n)-partite *where k is the number of blocks and $n = kB$ is the total input length.*

Example 5.2. Suppose that we have $k = 2$ blocks each of length $B = 3$ bits, with regularity parameter $b = 2$, and assume that $\hat{g} : \{0,1\}^3 \times \{0,1\}^3 \to \{0,1\}$ is the equality function that outputs 1 if and only if the first block, x_1, equals to the second block, x_2. In this case, the function $g : \{0,1\}^3 \times \{0,1\}^3 \to \{0,1\}$ is a 2-regular realization of $\hat{g}$ if for every input (x_1, x_2) of Hamming weight exactly $bk = 4$, the function g outputs 1 if and only if (1) both x_1 and x_2 have Hamming weight of exactly 2, and (2) $x_1 = x_2$. For inputs whose Hamming weight is not 4, the function g is unconstrained and may behave arbitrarily.

Example 5.3. Take $k = 2$, $B = 3$, and $b = 2$. Let $\hat{g}(x_1, x_2)$ be the threshold function that outputs 1 if and only if $\text{wt}(x_1 \circ x_2) \geq 4$. In this case, the slice function $g : \{0,1\}^3 \times \{0,1\}^3 \to \{0,1\}$ defined via

$$g(x_1, x_2) = 1 \qquad \iff \qquad \text{wt}(x_1) = \text{wt}(x_2) = 2 \quad \text{OR} \quad \text{wt}(x_1 \circ x_2) \geq 5$$

is a 2-regular realization of $\hat{g}$.

The following lemma is implicit in [38].

[10]Throughout this section we ignore rounding issues, and assume, for simplicity, that n is a multiple of B whenever it is needed.

[11]This means that a single specification can be realized by several different functions. For concreteness, one may assume that inputs of weight smaller than bk are rejected and inputs of weight larger than bk are accepted. Under this convention, g is also a slice function.

Lemma 5.4 (Realizing mid-part based on block-regular functions [38])**.** *Let $B = B(n)$ and $k = k(n)$ be integer parameters for which $k = n/B - 2\lceil \delta n \rceil$. Then, the function $f_{\mathrm{mid},\delta} : \{0,1\}^n \to \{0,1\}$ can be realized by a constant-depth formula of size $S = \frac{\binom{n}{n-\delta}\cdot\binom{k}{2\delta}}{\binom{n/k}{(\frac{n}{2}-\delta)/k}} 2^{(2\delta+1)n/k+O(\sqrt{n}\log n)}$ over AND,OR, and block regular gates with block-size B and number of blocks k.*[12]

We refer the reader to the original paper of [38] for a full proof of the lemma. For now, let us prove a weaker statement that will be also useful as a warmup for the next subsection.

Proposition 5.5 (Realizing middle-slice based on block-regular functions)**.** *Let n, B, k be integers such that $n = Bk$. Every $(n/2, n)$-slice function f can be realized by a constant-depth formula of size $S = nB^{O(k)}$ over threshold gates and block regular gates with regularity $b = B/2$, block-size B, and number of blocks k. Specifically, for $B = k = \sqrt{n}$, we derive a formula of sub-exponential size $S = 2^{\tilde{O}(\sqrt{n})}$.*

Sketch. Let $\Pi = (I_1, \dots, I_k)$ be a partition of $[n]$ to k subsets of size B each. An input $x \in \{0,1\}^n$ is *good* for the partition Π if for each $i \in [k]$ the sub-string $x[I_i] \in \{0,1\}^B$ is of Hamming weight exactly $b = B/2$. By a standard probabilistic argument, there exists a collection of $\ell = 2^{\tilde{O}(\sqrt{n})}$ partitions $\Pi_1, \dots, \Pi_\ell$ such that every n-bit string x whose Hamming weight is $n/2$ is good for at least one partition Π_j.

For each $i \in [\ell]$, partition the inputs of f to k block according to Π_i, denote the resulting function by $f_i : (\{0,1\}^{\sqrt{n}})^{\sqrt{n}} \to \{0,1\}$, and let g_i be a $B/2$-regular realization of f_i (that is, $g_i(x) = f(x)$ for all inputs x of weight $n/2$ such that x is good for Π_i and $g_i(x) = 0$ for all other inputs of weight $n/2$). By definition, it holds that

$$f(x) = \bigvee_{i\in[\ell]} g_i(x)$$

for every input x whose weight is exactly $n/2$. Since lighter inputs and heavier inputs can be easily handled via threshold gates, the proposition follows. $\square$

In fact, LV further reduce general b-regular functions $f : (\{0,1\}^B)^k \to \{0,1\}$ to 1-regular functions (i.e., k-partite functions). The reduction is based on "1-hot encoding" and therefore introduces an exponential blow-up in the block-length. Details follow.

Notation 5.6 (1-hot encoding)**.** *For a string $v \in \{0,1\}^B$, we define $1_v \in \{0,1\}^{2^B}$ as the weight 1 string whose coordinates are indexed by strings of length B (ordered according to the binary numbers they represent) and*

$$1_v[x] = \begin{cases} 1 & x = v \\ 0 & x \neq v. \end{cases}$$

Proposition 5.7 (Block-regular functions based on k-partite functions, implicit in [38])**.** *Every function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ admits a b-regular realization g that can be computed by a formula of size $O(kB2^B)$ over AND, OR gates, and a single $(k, N = k2^B)$-partite gate $f : (\{0,1\}^{2^B})^k \to \{0,1\}$.*

Proof. Take f to be the k-partite function that satisfies the equation

$$f(1_{x_1}, \dots, 1_{x_k}) = \hat{g}(x_1, \dots, x_k)$$

[12]More precisely, to cope with rounding issues, LV extend the notion of b-regular gates by relaxing the notion of b-regular strings to include every string $x = (x_1, \dots, x_k) \in (\{0,1\}^B)^k$ that each of the first blocks $x_1, \dots, x_\ell$ have Hamming weight of b and each of the other remaining blocks have Hamming weight of $b + 1$.

for every b-regular input $(x_1,\dots,x_k)$. Furthermore, assume, WLOG, that $f(y_1,\dots,y_k)=0$ whenever the input $(y_1,\dots,y_k)$ contains an all zero block, $y_i=0^{2^B}$.[13]

Consider the function g that maps an input $(x_1,\dots,x_k)\in(\{0,1\}^B)^k$ to a string $y=(y_1,\dots,y_k)\in(\{0,1\}^{2^B})^k$ and outputs $f(y_1,\dots,y_k)$, where for $i\in[k]$ the ith block $y_i\in\{0,1\}^{2^B}$ is defined via

$$y_i[v]=\begin{cases}\bigwedge_{j:v[j]=1} x_i[j], & \text{if } v\in\{0,1\}^B \text{ is of weight } b\\ 0, & \text{otherwise}\end{cases}.$$

Correspondingly, $y_i[v]=1$ iff $x_i\geq v$. In particular, if $\mathrm{wt}(x_i)=b$, then $y_i=1_{x_i}$. To see that g is a b-regular realization of $\hat{g}$, consider an input $x=(x_1,\dots,x_k)$ of total weight bk. If this input is b-regular, then f is applied over a 1-regular input $y=(1_{x_1},\dots,1_{x_k})$, and so the output agrees with $\hat{g}(x_1,\dots,x_k)$. If the input x has Hamming weight bk but does not respect the partition, then some block x_i must have Hamming weight smaller than b, and the corresponding block y_i equals to the all zero-string. By assumption, f outputs zero, as required. Finally, as each $y_i[v]$ is a formula of size at most B and there are $k2^B$ such formulas, g can be computed by a formula of size $O(kB2^B)$. $\square$

Example 5.8. Let $\hat{g}:\{0,1\}^3\times\{0,1\}^3\to\{0,1\}$ be the equality function, i.e., $\hat{g}(x_1,x_2)=1$ iff $x_1=x_2$. Let $f:\{0,1\}^8\times\{0,1\}^8\to\{0,1\}$ be the 2-partite function that outputs 1 on singletons of the form $(1_v,1_u)$ if and only if $u=v$. Assume, in addition, that $f(0^8,x)=f(x,0^8)=0$ for every x. Then the following function $g:\{0,1\}^3\times\{0,1\}^3\to\{0,1\}$ is a 2-regular realization of $\hat{g}$,

$$\begin{aligned}g(x_1[1],x_1[2],x_1[3],x_2[1],x_2[2],x_2[3]) := f(&0,0,0,x_1[2]\wedge x_1[3],0,x_1[1]\wedge x_1[3],x_1[1]\wedge x_1[2],0,\\ &0,0,0,x_2[2]\wedge x_2[3],0,x_2[1]\wedge x_2[3],x_2[1]\wedge x_2[2],0).\end{aligned}$$

For example, $y_1[011]$ – the 4th bit in the input of f – is replaced by $x_1[2]\wedge x_1[3]$.

Wrapping-up. By combining the LV decomposition Eq. (2) with Lemma 5.4 and Proposition 5.7, LV derive the following theorem.

Theorem 5.9. *Every monotone function $f:\{0,1\}^n\to\{0,1\}$ can be realized by a constant-depth formula of size $2^{0.994n+o(n)}$ over slice gates, and, specifically, over AND, OR, and partite gates.*

Applebaum et al. [2] further reduced the exponent by improving the formula size of the extreme parts, $f_{\mathrm{bot},\delta}$ and $f_{\mathrm{top},\delta}$, of f. Specifically, they showed, with the aid of combinatorial designs, that these functions can be realized by formulas over general monotone gates whose domain is smaller than n. As a result, one can replace the naive CNF/DNF implementations with a better formula that makes recursive calls to the LV formula. Evidently, this construction also gives rise to non-trivial formulas over AND, OR, and k-partite slices, as claimed.

Beimel and Farràs [7] observed that "typical" monotone functions behave almost like $(n/2,n)$-slices. Specifically, all but $o(1)$ fraction of all monotone functions are non-trivial only for inputs of weight $n/2\pm 1$ and take the constant value zero (resp. one) over lighter (resp. heavier inputs) [35]. By using a simplified version of the LV construction, they prove that all but $o(1)$-fraction of n-bit monotone functions can be realized by a secret-sharing scheme with share size $2^{\tilde{O}(\sqrt{n})}$. Following their work, we conclude the following theorem (that can be derived from Lemma 5.4 by plugging $\delta(n)=o(n/\log n)$ and $k(n)=\sqrt{\frac{n\delta(n)}{\log n}}$).

Theorem 5.10. *Almost all monotone functions over n bits can be computed by a constant depth formula over slice gates of size $2^{\tilde{O}(\sqrt{n})}$.*

[13]If this condition does not hold, replace f with the formula $f(y)\bigwedge\left(y_1[1]\vee\cdots\vee y_1[2^B]\right)\bigwedge\cdots\bigwedge\left(y_k[1]\vee\cdots\vee y_k[2^B]\right)$.

5.2 The ABNP Construction and Its Follow-Ups

The ABNP construction [3] further reduces the overall exponent by deriving a smaller formula for $f_{\mathrm{mid},\delta}$, the middle part of f. To this end, they introduce the following relaxed variant of block-regular functions.

Definition 5.11 (($a : b$)-somewhat-regular functions)**.** *We say that a string $x = (x_1, \dots, x_k) \in (\{0,1\}^B)^k$ is* $(a : b)$-regular *if each block x_i, $i \in [k]$, has Hamming weight of at least a and at most b.*

A function $g : (\{0,1\}^B)^k \to \{0,1\}$ is an $(a : b)$-somewhat-regular *realization of some monotone function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ if g agrees with $\hat{g}$ on all inputs that $(a : b)$-respect the input partition. The function g can take arbitrary values on* all *other inputs. When $\hat{g}$ is unspecified we refer to g as $(a : b)$-somewhat-regular, or, simply $(a : b)$-regular.*

Unlike the case of block-regular functions, here we make no requirements on inputs that are not $(a : b)$-regular.

Example 5.12. Take $k = 2$, $B = 3$ and assume that $\hat{g}(x_1, x_2)$ is the equality function that outputs 1 if and only if the strings $x_1, x_2 \in \{0,1\}^3$ are equal. In this case, the function $g : \{0,1\}^3 \times \{0,1\}^3 \to \{0,1\}$ is a $(2, 3)$-regular realization of $\hat{g}$ if it is constrained over every input (x_1, x_2) where $x_i \in \{0,1\}^3$ is of Hamming weight 2 or 3. For such inputs, the output $g(x_1, x_2)$ should be 1 if and only if $x_1 = x_2$. For all other inputs (e.g., when the weight of x_1 is 1 and x_2 is an arbitrary string), the function g is unconstrained and may behave arbitrarily.

One can use block-regular functions to realize so-called $(a : b, n)$ *multislices*. The latter functions are monotone functions over n bits that can take arbitrary values (subject to the monotonicity constraint) over inputs of weight larger than a and smaller than b, and take the constant value 0 (resp., 1) on lighter (resp. heavier) inputs.

Proposition 5.13 (Realizing multislices based on somewhat-regular functions [3, 4])**.** *Every $(a : b, n)$ multislice function f can be realized by a constant-depth formula F over threshold gates and $(a' : b')$-regular gates over over n bits with $k = \sqrt{n}$ blocks and thresholds of $a' = a/\sqrt{n} - n^{\varepsilon}$ and $b' = b/\sqrt{n} + n^{\varepsilon}$ for some constant $\varepsilon < 1/2$. The formula F uses $2^{o(n)}$ threshold gates and $O(n)$ somewhat-regular gates.*

Sketch. Fix some proximity parameter $\varepsilon > 0$. We prove the proposition under the assumption that $n^{\delta} \leq a \leq b \leq n - n^{\delta}$ for some $\delta \in (0, 1)$ that depends on ε. If a is smaller or b is larger than the "missing" area can be handled by appending a CNF or DNF to the construction with an additive sub-exponential overhead.

Let $\Pi = (I_1, \dots, I_k)$ be a partition of $[n]$ to $k = \sqrt{n}$ subsets of size $n/k = \sqrt{n}$ each. An input $x \in \{0,1\}^n$ is *good* for the partition Π if for each $i \in [k]$, the sub-string $x[I_i] \in \{0,1\}^{\sqrt{n}}$ is of Hamming weight at least $a/\sqrt{n} - n^{\varepsilon}$ and at most $b/\sqrt{n} + n^{\varepsilon}$. By a standard probabilistic argument (e.g., [4, Lemma B.4]), there exists a collection of $\ell = O(n)$ partitions $\Pi_1, \dots, \Pi_{\ell}$ such that every n-bit string x whose Hamming weight is in $[a, b]$ is good for a majority of the partitions.

For each $i \in [\ell]$, partition the inputs of F to k block according to Π_i, denote the resulting function by $F_i : (\{0,1\}^{\sqrt{n}})^{\sqrt{n}} \to \{0,1\}$, and let g_i be an $(a' : b')$-somewhat-regular realization of F_i. Let x be an input whose weight is in (a, b). If x is good for a partition Π_i, then $g_i(x) = f(x)$, otherwise we have no guarantee on $g_i(x)$. As x is good for is good for the majority of the partitions,

$$F(x) = \mathrm{MAJ}_{i \in [\ell]}\, g_i(x).$$

Since lighter inputs and heavier inputs can be easily handled via threshold gates, the proposition follows. □

In [3] it is shown how to further realize somewhat-regular functions by a formula over partite slice gates.

Theorem 5.14 (Realizing somewhat-regular functions based on partite functions)**.** *Let B, k, $0 \le a' \le b' \le B$ be some integers and let $n = Bk$. Every monotone function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ admits an $(a' : b')$-somewhat-regular realization g that can be computed by a constant-depth formula over AND, OR, and $(\sqrt{n}, 2^{\sqrt{n}})$-partite gates of size*

$$G \le 2^{O(2B+k\log k)} \cdot (t\,\mathrm{polylog}(t))^k \quad \textit{where } t = \sum_{i=a'}^{b'} \binom{b'}{i}.$$

The proof of the theorem is deferred to the following subsections. For now, let us combine Theorem 5.14 with Proposition 5.13 and record the following corollary. Below we let $\mathrm{H}_2(\cdot)$ denote the binary entropy function.

Corollary 5.15. *Let $0 \le \alpha \le \beta \le 1$ be real numbers. Every $(\alpha n : \beta n, n)$ multislice function F can be realized by an S-size formula over threshold gates and $(\sqrt{n}, 2^{\sqrt{n}})$-partite gates where $S \le 2^{\beta n + o(n)}$ regardless of the value of α, and $S \le 2^{\beta\,\mathrm{H}_2(\alpha/\beta)n+o(n)}$ when $\alpha > \beta/2$.*

Proof. Let $B = k = \sqrt{n}$. First observe that in Theorem 5.14 , it holds that $t = \sum_{i=a'}^{b'} \binom{b'}{i} \le 2^{b'}$ and that, for $a' \in (b'/2, b')$, a tighter-bound of $O(b')\binom{b'}{a'} \le 2^{b'\,\mathrm{H}_2(a/b')+o(n)}$ holds. Thus, by plugging Theorem 5.14 into Proposition 5.13, with $a' = \alpha B - o(B)$ and $b' = \beta B + o(B)$, we can upper-bound S by $2^{o(n)} \cdot (t\,\mathrm{polylog}(t))^k$, which is $2^{kB\beta+o(n)} = 2^{\beta n+o(n)}$ or, when $\alpha > \beta/2$, by $2^{kB\beta\,\mathrm{H}_2(\alpha/\beta)+o(n)} = 2^{\beta\,\mathrm{H}_2(\alpha/\beta)n+o(n)}$, as required. □

General monotone functions based on upslices. Corollary 5.15 can be used to realize the mid-part function in the LV-decomposition (or better yet in its optimized variant [2]). Indeed, this approach was taken in [3] leading to a FOS of size $2^{0.637n+o(n)}$ for general monotone functions. We describe an alternative approach from [4] that leads to the best known exponent. The construction starts by realizing so-called (b,n)-*downslices* which are simply monotone CNFs whose clauses each contain exactly $n-b$ variables. Observe that a naive realization of such functions via a CNF results in a formula of size $2^{\mathrm{H}_2(b/n)n}$. The following proposition shows that one can do much better based on partite gates.

Definition 5.16. *A monotone function $f : \{0,1\}^n \to \{0,1\}$ is a (b,n)-downslice if all its* maxterms *are of size exactly b.*

Thus an (b,n)-downslice is unconstrained over b-weight inputs, takes the value 1 on heavier inputs, and (unlike slice functions) evaluates to 0 on an input y of weight smaller than b only if there exists a *larger* input $x \ge y$ of weight exactly b on which the function evaluates to 0.

Proposition 5.17 ([4])**.** *Every $(b = \beta n, n)$-downslice can be realized by a constant-depth formula over threshold gates and $(\sqrt{n}, 2^{\sqrt{n}})$-partite gates of size*

$$S \le \begin{cases} 2^{\beta n+o(n)}, & \textit{if } \beta \le 1/2 \\ 2^{(\mathrm{H}_2(\beta)-(1-\beta))n+o(n)}, & \textit{if } \beta > 1/2 \end{cases}.$$

Sketch. The case of $\beta \le 1/2$ follows immediately from Corollary 5.15. Indeed, every (b,n)-downslice is also a $(0 : b, n)$-multislice.

For $\beta > 1/2$, we show that f can be realize by a formula F of size $2^{(\mathrm{H}_2(\beta)-2(1-\beta))n+o(n)}$ that makes use of AND gates and a single bottom layer of $(n-b, 2(n-b))$-downslice gates. Since each of these gates can be realized by a formula of size $2^{1/2\cdot 2(1-\beta)n+o(n)} = 2^{(1-\beta)n+o(n)}$ (as shown for the $\beta \le 1/2$ case), we derive a formula of size

$$2^{(\mathrm{H}_2(\beta)-2(1-\beta))n+o(n)} \cdot 2^{(1-\beta)n+o(n)} = 2^{(\mathrm{H}_2(\beta)-(1-\beta))n+o(n)},$$

as required.

We proceed by describing the formula F. Pick a family $\mathcal{S} = \{S_i\}_{i=1}^{L}$ of sets of size $2b-n$ such that every b-subset of $[n]$ is a super-set of at least one set of $\mathcal{S}$, that is, for every $B \subseteq \{1,\dots,n\}$ of size b there exists at least one $S_i \in \mathcal{S}$ such that $S_i \subset B$. By standard probabilistic argument, one can get such a family $\mathcal{S}$ of size $L \le 2^{(\mathrm{H}_2(\beta)-2(1-\beta))n+o(n)}$. Denote by $\overline{S}_i := [n]\backslash S_i$ the complement of S_i. Next, for every $S_i \in \mathcal{S}$ define the function $f_i : \{0,1\}^{2n-2b} \to \{0,1\}$ that given an input $x' \in \{0,1\}^{2n-2b}$ maps x' to an n-bit string x where $x[S_i] = 1^{|S_i|}$ and $x[\overline{S}_i] = x'$, and outputs $f(x)$.[14] Finally let the formula $F : \{0,1\}^n \to \{0,1\}$ be defined as

$$F(x) = \bigwedge_{i=1}^{L} f_i(x[\overline{S}_i]).$$

We show that F computes f. First, assume that x is accepted by f, and fix some $i \in [L]$. To see that $f_i(x) = 1$, recall that $f_i(x) = f(x_{S_i\to 1})$ where $x_{S_i\to 1}$ stands for the n-bit string that is identical to x except that its S_i coordinates are taken to be ones. Therefore, $x_{S_i\to 1} \ge x$ and by monotonicity $f(x_{S_i\to 1}) \ge f(x) = 1$, as required. Next, we show that every maxterm x of f is being rejected by some f_i. Since f is a b-downslice, we may assume that x is of weight b, and denote its support by B. By the covering property of $\mathcal{S}$, there exists an index $i \in [L]$ such that $S_i \subseteq B$. Thus, $x_j = 1$ for every $j \in S_i$ and $x_{S_i\to 1} = x$. By definition, $f_i(x[\overline{S}_i]) = f(x_{S_i\to 1})$, and $f(x_{S_i\to 1}) = f(x) = 0$, as required.

Finally, we show that each f_i is a $(n-b, 2(n-b))$-downslice. Fix some maxterm $x' \in \{0,1\}^{2n-2b}$ of f_i. We show that x' has Hamming weight of exactly $n-b$ by showing that its n-bit extension x, where $x[S_i] = 1^{|S_i|}$ and $x[\overline{S}_i] = x'$, is a maxterm of f. By definition, x is a 0-input of f. Moreover, every $y > x$ must be a 1-input of f. (Otherwise, x' cannot be a maxterm of f_i.) Since any maxterm of f is of size b, the maxterms of f_i are of size $b-(2b-n) = n-b$ as required. □

Observe that every monotone function f can be written as a conjunction $\bigwedge_{b=1}^{n} f_b$, where f_b is the (b,n)-downslice that contains all b-weight maxterms of f. By Proposition 5.17, each of these downslices can be realized by a formula of size at most $1.5^{n+o(n)}$ (the maximal exponent is obtained for $\beta = 2/3$). Hence, f can be realized by a FOS of size $1.5^{n+o(n)}$, and Theorem 1.1 follows.

In the remaining parts of this section (Section 5.2.1 and Section 5.2.2) we prove Theorem 5.14.

5.2.1 Somewhat-regular gates based on robust partite gates

In order to realize $(a : b)$-somewhat-regular gates, Applebaum et al. [3] introduce the notion of *robust-CDS* protocols. Recall that CDS protocols are just secret-sharing schemes for partite functions. The robust version of this notion asserts that the secret-sharing remains secure for certain bigger coalitions. On a first glance, it is not clear how to translate this property to the formula setting. Nevertheless, we show that the concrete immunization tools that were presented by Applebaum et al. [3] can be captured as formula manipulations. In the remaining subsections, we sketch this interpretation and show how to realize $(a : b)$-somewhat-regular gates via partite gates. We will need the following strengthening of k-partite functions as an intermediate notion.

[14]For a set $S = \{i_1,\dots,i_\ell\}$ and a string $x \in \{0,1\}^n$, we denote $y = x[S]$ as the ℓ-bit sting where $y_j = x_{i_j}$.

Definition 5.18 (t-robust k-partite functions)**.** *Let $f' : (\{0,1\}^A)^k \to \{0,1\}$ be a k-*partite *function. We say that $f : (\{0,1\}^A)^k \to \{0,1\}$ is a t-*robust *realization of f' if for every input $(y_1, \dots, y_k)$, where y_i is of Hamming weight at most t for every $i \in [k]$, it holds that $f(y) = 1$ if and only if there exists a 1-regular* witness *$y^* \le y$ (i.e., $y^* = (y^*_1, \dots, y^*_k)$ such that $\mathrm{wt}(y^*_i) = 1$ for $1 \le i \le k$) for which $f'(y^*) = 1$. The value of f over all other inputs can be arbitrary.*

Recall that a k-partite function f' is specified only on inputs for which each block contains a single "1". The t-robust version f of f' acts similarly on these inputs with an additional restriction over inputs for which each block has at most t ones. Over this set of inputs, f should act as a monotone function whose only min-terms are the 1-regular strings that are accepted by f'.

Example 5.19. Let $f' : \{0,1\}^8 \times \{0,1\}^8 \to \{0,1\}$ be the 2-partite function that outputs 1 on singletons of the form $(1_v, 1_u)$ if and only if $u = v$. Then, the function $f : \{0,1\}^8 \times \{0,1\}^8 \to \{0,1\}$ is a 3-robust realization of f' if f is constrained over every input (y_1, y_2) for which y_1 and y_2 are non-zero strings of weight at most 3. On such inputs, the output $f(y_1, y_2)$ takes the value 1 if and only if there exist a weight 1 witnesses $y^*_1 \le y_1$ and $y^*_2 \le y_2$ that are accepted by f', namely, if y_1 and y_2 have both "1" in some common location $i \in [8]$. For all other inputs, f is unconstrained and may behave arbitrarily. Note that f' is unconstrained on a smaller set of inputs (i.e., only on the 1-regular inputs) and may act arbitrarily on, say, heavier strings.

By using 1-hot encoding, we can realize somewhat-regular functions based on robust partite functions while introducing an exponential blow-up in the block length. The transformation is analogous to the one described in Proposition 5.7, and the reader may safely skip the proof.

Proposition 5.20 (Realizing somewhat-regular functions based on robust partite functions)**.** *Every monotone function $\hat{g} : (\{0,1\}^B)^k \to \{0,1\}$ admits an $(a : b)$-somewhat-regular realization g that can be computed by a constant-depth formula of size $O(kB2^B)$ over AND gates and a single t-robust k-partite functions $f : (\{0,1\}^{2^B})^k \to \{0,1\}$ of total input length of $N = k2^B$, where $t = \sum_{i=a}^{b} \binom{b}{i}$.*

Sketch. Take f' to be the k-partite function that satisfies the equation

$$f'(1_{x_1}, \dots, 1_{x_k}) = \hat{g}(x_1, \dots, x_k)$$

for every $(a : b)$-regular input $(x_1, \dots, x_k)$, and let f be a t-robust realization of f'.

Consider the function g that maps an input $(x_1, \dots, x_k) \in (\{0,1\}^B)^k$ to a string $y = (y_1, \dots, y_k) \in (\{0,1\}^{2^B})^k$ and outputs $f(y_1, \dots, y_k)$, where for every $i \in [k]$ and $v \in \{0,1\}^B$ of Hamming weight in $[a, b]$, we set $y_i[v] = \bigwedge_{j:v[j]=1} x_i[j]$, and for every other $v \in \{0,1\}^B$ we set $y_i[v] = 0$. That is, to compute the bit $y_i[v]$ we take the conjunction of all the $x_i[j]$'s for which $v[j] = 1$.

By construction, g can be written as a formula of size $O(kB2^B)$ that employs a single t-robust k-partite gate. Furthermore, $g(x_1, \dots, x_k)$ forms an $(a : b)$-regular realization of $\hat{g}$, as required. Indeed, any $(a : b)$-regular input $x = (x_1, \dots, x_k)$ for g induces an f-input $y = (y_1, \dots, y_k)$ such that every string y_i has an Hamming weight of at most $t = \sum_{i=a}^{b} \binom{b}{i}$. This means that y is accepted by g if and only if there exists some 1-regular $y^* \le y$ that is being accepted by f'. The latter happens if and only if the vector x^* for which $y^* = (1_{x^*_1}, \dots, 1_{x^*_k})$ is being accepted by $\hat{g}$. By the monotonicity of $\hat{g}$, this happens if and only if $\hat{g}(x) = 1$. □

5.2.2 From k-partite functions to t-robust k-partite functions

We show how to take a (monotone) k-partite function $f' : (\{0,1\}^A)^k \to \{0,1\}$ and turn it into a *t-robust* function $f : (\{0,1\}^A)^k \to \{0,1\}$. That is, we construct a "small" monotone formula for f that makes use of

f' gates and standard AND, OR gates. Recall that f' is constrained only on 1-regular inputs. For simplicity (and essentially without loss of generality), let us assume that f' is a slice function that accepts all heavier inputs, and in particular, f' is monotone. In contrast to f', the function f is constrained also for inputs y that have at most t ones in each block (hereafter referred to as *$(\leq t)$-regular* inputs). Specifically, f should accept such inputs if and only if there exists a 1-regular witness $y^* \leq y$ that is accepted by f'. The main challenge is therefore to filter-out and reject $(\leq t)$-regular inputs that have no witness, despite the fact that such inputs are accepted by the non-robust function f'.

The high-level idea is to take an input $y = (y_1, \dots, y_k)$, randomly zero-out some of the bits, and feed the resulting string, y', into an f' gate. Crucially, $y' \leq y$, and therefore, if the input y does not satisfy f', then y' still does not satisfy it, and no harm was done. Consider an input y that satisfies f' for which there is no 1-regular witness $y^* \leq y$ that is being accepted by f'. If we zero-out enough coordinates, then we are likely to generate low-weight y' that is not satisfied by f' and so we gain "robustness". Of course, we cannot be too aggressive, because we may reject inputs y that should be accepted (i.e., for which there exists a 1-regular accepting witness $y^* \leq y$).

The actual construction uses many copies of f' while randomly choosing which bits to zero. There are several ways to instantiate this idea (see [3]) and here we use a simplified variant from [8]. We need the following notation: For a set $X \subset [A]$ and a string $y \in \{0,1\}^A$, let $y(X)$ denote the A-bit string that agrees with y over all indices in the set X and takes the value zero in all other entries.

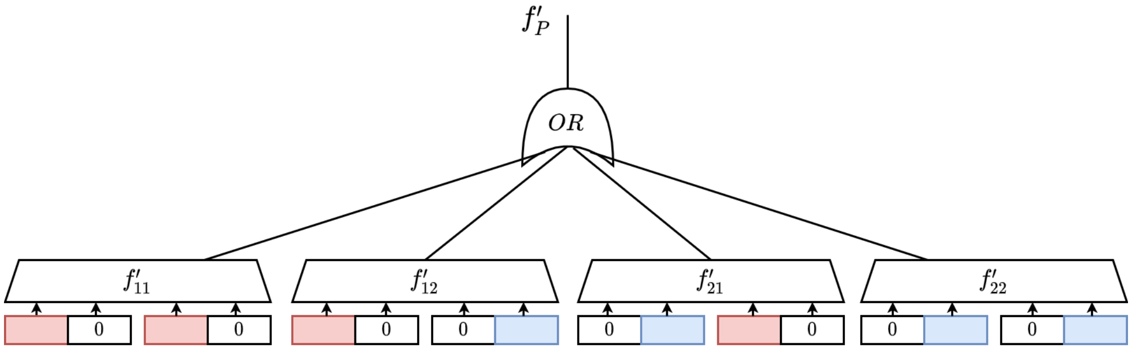


Figure 2: An illustration of Construction 5.21 for a 2-robust realization of a 2-partite function $f' : \{\{0,1\}^A\}^2 \to \{0,1\}$. Given a partition P of A to a "left" half and a "right" half, we define the function f'_P to be the disjunction of the functions $f'_{1,1}, f'_{1,2}, f'_{2,1}, f'_{2,2}$, in which different parts of the input are zeroed-out.

Warm-up. To illustrate the ideas of the construction, we construct a 2-robust realization of a 2-partite function $f' : (\{0,1\}^A)^2 \to \{0,1\}$. The construction is also depicted in Figure 2. Consider a partition (X_1, X_2) of $[A]$. For every $\alpha_1, \alpha_2 \in \{1,2\}$ define a function f'_{α_1,α_2}, where

$$f'_{\alpha_1,\alpha_2}(y_1, y_2) := f'(y_1(X_{\alpha_1}), y_2(X_{\alpha_2})),$$

that is, the function zeroes-out all the inputs in y_i outside the set X_{α_i} for $i \in \{1,2\}$.

Consider an input for which the realization should output 1, that is, an input $y = (y_1, y_2)$ such that $\mathrm{wt}(y_i) \leq 2$ for $i = 1,2$ and there exists some witness 1-regular witness $y^* = (y_1^*, y_2^*)$ such that $y^* \leq y$ and $f'(y^*) = 1$. Let i_1 (resp. i_2) be the unique index such that $y_1^*[i_1] = 1$ (resp. $y_2^*[i_2] = 1$). As (X_1, X_2) is a partition of A, there are α_1, α_2 such that $i_1 \in X_{\alpha_1}$ and $i_2 \in X_{\alpha_2}$. Thus, $y_1(X_{\alpha_1}) \geq y_1^*$ and $y_2(X_{\alpha_2}) \geq y_2^*$. By the monotonicity of f' we conclude that

$$f'_{\alpha_1,\alpha_2}(y_1, y_2) = f'(y_1(X_{\alpha_1}), y_2(X_{\alpha_2})) \geq f'(y_1^*, y_2^*) = 1.$$

Thus,

$$\bigvee_{\alpha_1,\alpha_2 \in \{0,1\}} f'_{\alpha_1,\alpha_2}(y) = 1 \quad \text{for an input } y \text{ such that the realization should output 1.} \tag{3}$$

Now consider an input for which the realization should output 0, that is, an input such that $\mathrm{wt}(y_1) = \mathrm{wt}(y_1) = 2$ and

$$f'(y^*) = 0 \ \text{ for every 1-regular } y^* \leq y \tag{4}$$

(the cases where $\mathrm{wt}(y_1) < 2$ or $\mathrm{wt}(y_2) < 2$ are similar). Let i_1, i'_1, i_2, i'_2 be the indices such that $y_1[i_1] = y_1[i'_1] = y_2[i_2] = y_2[i'_2] = 1$. Assume that P is "lucky" for y in the following sense:

$$|X_1 \cap \{i_1, i'_1\}| = 1 \text{ and } |X_2 \cap \{i_2, i'_2\}| = 1,$$

that is, exactly one of the indices in which y_1 is one is in X_1 and the other index is in X_2; this also holds for y_2. In this case, $\mathrm{wt}(y_1(X_1)) = \mathrm{wt}(y_1(X_2)) = \mathrm{wt}(y_2(X_1)) = \mathrm{wt}(y_2(X_2)) = 1$ and by (4) we get that $f'_{\alpha_1,\alpha_2}(y) = f'(y_1(X_{\alpha_1}), y_2(X_{\alpha_2})) = 0$ for every α_1, α_2. Thus,

$$\bigvee_{\alpha_1,\alpha_2 \in \{0,1\}} f'_{\alpha_1,\alpha_2}(y) = 0 \quad \text{for an input } y \text{ such that the realization should output 0 and } P \text{ is "lucky" for } y. \tag{5}$$

However, if P is not "lucky" and, for example, $|X_1 \cap \{i_1, i'_1\}| = 2$, then $\mathrm{wt}(y_1(X_1)) = 2$ and we do not have any guarantees on $f'_{\alpha_1=1,\alpha_2}(y)$.

To overcome the above problem, we randomly choose $X_1 \subseteq [A]$ with uniform distribution and define $X_2 = A \backslash X_1$. With probability $1/4$ the partition (X_1, X_2) is "lucky" for a given y. If we choose $m = c \cdot \log n$ random partitions $P_1, \ldots, P_m$ (for a big enough constant c) then the probability that at least one partition P_i is "lucky" for y is greater than $1 - 1/n^4$. This implies that there are partitions $P_1 = (X_1^1, X_2^1), \ldots, P_m = (X_1^m, X_2^m)$ such that for *every* input y for which the realization should output 0 there is at least one partition P_i that is "lucky" for y. For these partitions we define

$$f'_{P_i}(y_1, y_2) := \bigvee_{\alpha_1,\alpha_2 \in \{0,1\}} f'(y_1(X^i_{\alpha_1}), y_2(X^i_{\alpha_2})),$$

and

$$f'_{P_1,\ldots,P_m}(y_1, y_2) := \bigwedge_{1 \leq i \leq m} f'_{P_i}(y_1, y_2).$$

By (3), (5), and the choice of $P_1, \ldots, P_m$, we get that $f'_{P_1,\ldots,P_m}$ is a 2-robust realization of f'. Furthermore, $f'_{P_1,\ldots,P_m}$ can be computed by a formula of size $O(\log n)$ over f' gates and AND, OR gates

Construction 5.21 of a t-robust realization of a k-partite function is similar to the warm-up. However, for efficiency reasons, we partition A to $O(kt^2)$ sets.

Construction 5.21. *For a partition $P = (X_1, \ldots, X_\ell)$ of the set $[A]$, and a sequence $(\alpha_1, \ldots, \alpha_k) \in [\ell]^k$, let $f'_{\alpha_1,\ldots,\alpha_k}$ be the following function*

$$f'_{\alpha_1,\ldots,\alpha_k}(y_1, \ldots, y_k) := f'(y_1(X_{\alpha_1}), \ldots, y_k(X_{\alpha_k})),$$

Furthermore, define the function

$$f'_P(y_1,\dots,y_k) := \bigvee_{\alpha_1,\dots,\alpha_k\in[\ell]^k} f'_{\alpha_1,\dots,\alpha_k}(y_1,\dots,y_k).$$

(Observe that f'_P can be computed by using ℓ^k copies of f'.)
Given m partitions $\mathcal{P}=(P_1,\dots,P_m)$ of the set $[A]$ we let

$$f'_{\mathcal{P}}(y_1,\dots,y_k) := \bigwedge_{1\le i\le m} f_{P_i}(y_1,\dots,y_k).$$

Analysis. Let us begin by analyzing the function f'_P for some partition $P=(X_1,\dots,X_\ell)$. Fix an $(\le t)$-regular input $y=(y_1,\dots,y_k)$. We examine two cases for y.

- Consider an input y for which a t-robust realization of f' should output one, that is, there exists a 1-regular witness $y^*\le y$ for which $f'(y^*)=1$. Recall that each block of y^* is a standard unit vector of length A, and therefore, for each block $i\in[k]$, there is a unique $\alpha_i\in[\ell]$ for which $y^*_i(X_{\alpha_i})=y^*_i$. (That is, the single non-zero entry of y^*_i falls into the set X_{α_i}.) Thus, as we assume that f' is monotone,

$$f'_{\alpha_1,\dots,\alpha_k}(y_1,\dots,y_k)\ge f'_{\alpha_1,\dots,\alpha_k}(y^*_1,\dots,y^*_k)=f'(y^*_1,\dots,y^*_k)=1$$

  and $f'_P(y_1,\dots,y_k)=1$ as we would like.

- Consider an input y for which a t-robust realization of f' should output zero, that is, $f'(y^*)=0$ for every $y^*=(y^*_1,\dots,y^*_k)$ such that $y^*\le y$ and $\mathrm{wt}(y^*_i)\le 1$ for every $i\in[k]$. In this case it is possible that $f'_P(y)=1$. However, if for every $(\alpha_1,\dots,\alpha_k)$ and every block y_i we have zeroed-out all bits in which $y_i[j]=1$ except for at most 1 in $f'_{\alpha_1,\dots,\alpha_k}$ (i.e., $\mathrm{wt}(y_i(X_\alpha))\le 1$ for every $1\le i\le k$ and $1\le\alpha\le\ell$), then $f'_{\alpha_1,\dots,\alpha_k}(y)$ computes f' on an input z such that $f'(z)=0$ and $f'_P(y)=0$. Put differently, $f'_P(y)=f(y)$ as long as y does not 2-*collide* over the partition P. Here we say that a string $y\in(\{0,1\}^A)^k$ r-*collides* over P if there exists an index $i\in[k]$ and a set $\alpha\in[\ell]$, such that the string $y_i[X_\alpha]$ has Hamming weight of at least r.

Fix some $(\le t)$-regular y, and consider a random ℓ-partition $P=(X_1,\dots,X_\ell)$ of A where every $i\in A$ is placed in the set X_{j_i} for uniformly and independently chosen $j_i\in[\ell]$. For $\ell=O(kt^2)$, the probability that y 2-collides over P is at most $1/2$. This implies that there are $m=O(Ak)$ partitions $\mathcal{P}=\{P_1,\dots,P_m\}$ such that every $(\le t)$-regular string y does not 2-collide over at least one partition in $\mathcal{P}$. Thus, the function $f'_{\mathcal{P}}$ outputs the correct value for every $(\le t)$-regular input $y=(y_1,\dots,y_k)$. Overall, we derived a t-robust realization of f', which, by construction, can be computed by a formula of size $O(m\ell^k)=O(A\cdot O(k)^{k+1}\cdot t^{2k})$ over f' gates and AND, OR gates.

An optimization via a 2-level construction. The cost of the above robust realization of f' is still too expensive for our needs, and we would like to reduce the factor t^{2t} to t^t. Such a saving can be achieved as follows. First, we take $t'=\log t$, and use the above approach to obtain a t'-robust realization of f' by a formula F of size $O(A\cdot O(k)^{k+1}\cdot t'^{2k})$. Next, we apply Construction 5.21 except that the f'-gates are replaced with F-gates. Let $\mathcal{P}=\{P_1,\dots,P_m\}$ be the underlying collection of ℓ-partitions. A straightforward generalization of the previous analysis shows that $F_{\mathcal{P}}$ is a t-robust realization of f' if for every $(\le t)$-regular string $y \in (\{0,1\}^A)^k$ there exists a partition P_i such that y does not t'-collide over P_i.[15] By a simple analysis, the probability that some fixed $(\leq t)$-regular string y t'-collides over a randomly selected partition to $\ell = 2kt$ sets is at most $1/2$. Thus, there exists a collection of $m = O(Ak)$ ℓ-partitions $\mathcal{P} = \{P_1, \dots, P_m\}$ such that every $(\leq t)$-regular input $y = (y_1, \dots, y_k)$ does not t'-collide over at least one partition P_i. The function $F_{\mathcal{P}}$ forms a t-robust realization of f' that, by construction, can be computed by a formula of size $O(m\ell^k|F|) = O(A^2k^{O(k)}(t\log^2 t)^k)$ over f' gates and AND, OR gates. Overall, we derive the following proposition.

Proposition 5.22. *Let $f' : (\{0,1\}^A)^k \to \{0,1\}$ be a k-partite function and let t be an integer. Then, there is an implementation of a t-*robust *realization of f' by a constant-depth formula over AND, OR, and f' gates of total size of $O(A^2 \cdot k^{O(k)} \cdot (t\log^2(t))^k)$.*

Together with Proposition 5.20, this completes the proof of Theorem 5.14.

6 Secret Sharing from FOS for Long Secrets – Proof of Theorem 1.3

Suppose that the function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a FOS of size $2^{o(n/\log n)}$ over slice gates of weight bounded by $\mathrm{poly}(n)$. In this section, we prove that for sufficiently long secrets s, the function f can be realized with share size $2^{o(n)} \cdot |s|$ (i.e., it has a secret-sharing scheme with information ratio $2^{o(n)}$).

As a first step, we show that every FOS F of size S can be balanced into a FOS F' of depth $D = O(\log S)$ and size $S' = O(\mathrm{poly}(S))$ over slice gates with similar fan-in. The following theorem provides a more general statement that applies to any monotone formula over unbounded fan-in gates. We note that when all gates have fan-in 2 the following technique is very similar to the one used by Spira [50], with a different trade-off between the depth and size of the balanced formula.

Lemma 6.1 (Balancing monotone formulas over unbounded gates)**.** *Let F be a monotone formula of size S with Boolean gates of unbounded fan-in computing a monotone function f. Then, there is a monotone formula F' of depth $O(\log S)$ and size $O(\mathrm{poly}(S))$ computing the same function f. The gates of F' are the gates of F, AND gates, and OR gates.*

Proof. We construct the balanced formula F' recursively, where in each step we identify a gate in the formula F that divides F into sub-formulas of at most half the size of F, and continue recursively to balance each of these sub-formulas. The depth of the balanced formula F' is at most $3\log\mathrm{size}(F)$, as in each step we add at most 3 to the depth of F', where the number of steps in the recursion is at most $\log\mathrm{size}(F)$.

We denote $F = G(F_1, \dots, F_k)$, where G is the root-gate of F and $F_1, \dots, F_k$ are the sub-formulas rooted at the children of G.

We start with the simple case in which $\forall i \in [k] : \mathrm{size}(F_i) \leq \frac{\mathrm{size}(F)}{2}$. In this case, we continue to balance each sub-formula F_i recursively to an equivalent formula F'_i of depth $3\log|F_i|$ and output $F' = G(F'_1, \dots, F'_k)$, and by induction we get that

$$\mathrm{depth}(F') = 1 + \max_{i\in[k]} \mathrm{depth}(F'_i) \leq 1 + \max_{i\in[k]} 3\log\mathrm{size}(F_i) \leq 1 + 3\log\frac{\mathrm{size}(F)}{2} \leq 3\log\mathrm{size}(F).$$

Otherwise, there exists a sub-formula F_i such that $\mathrm{size}(F_i) \geq \frac{\mathrm{size}(F)}{2}$. Thus, we find a gate g in the formula F such that

[15]Indeed, if y does not t'-collide over some P_i and $f(y) = 0$, it must be the case that $F(y) = 0$ since F is t'-robust. It follows that $F_{\mathcal{P}}(y) = 0$, as required.

1. The size of the sub-formulas rooted at g is at least $\frac{\mathrm{size}(F)}{2}$, and

2. The size of each of the sub-formulas $H_1, \dots, H_\ell$ rooted at the children of g is less than $\frac{\mathrm{size}(F)}{2}$.

We can simply find such a gate g by traversing the formula F starting from the root G and choosing the child whose sub-formula is of size at least $\frac{\mathrm{size}(F)}{2}$. If such a child does not exist, we have found g.

For $b \in \{0,1\}$, let $\hat{F}_b$ be the formula F where we replace the sub-formula rooted at g by the constant b. The value of g selects if the formula F outputs $\hat{F}_0$ or $\hat{F}_1$, that is,

$$f = (\hat{F}_0 \wedge \overline{g(H_0, \dots, H_\ell)}\,) \vee (\hat{F}_1 \wedge g(H_0, \dots, H_\ell)).$$

As noted by [56], for a monotone formula, if $\hat{F}_0(x) = 1$ then $\hat{F}_1(x) = 1$ and $f(x) = 1$ regardless of the value of $g(H_0, \dots, H_\ell)$. This implies that f can be expressed by the monotone formula

$$f = \hat{F}_0 \vee (\hat{F}_1 \wedge g(H_0, \dots, H_\ell)).$$

Notice that $\mathrm{size}(\hat{F}_b) \le \frac{\mathrm{size}(F)}{2}$ for $b \in \{0,1\}$ and $\mathrm{size}(\hat{H}_i) \le \frac{\mathrm{size}(F)}{2}$ for $i \in [\ell]$. Thus, we construct the balanced formula F' by recursively balancing the formulas $\hat{F}_0$ and $\hat{F}_1$ and getting balanced formulas $\hat{F}'_0$ and $\hat{F}'_1$ respectively, recursively balancing the sub-formulas $H_1, \dots, H_\ell$ and getting balanced sub-formulas $H'_1, \dots, H'_\ell$ respectively, and letting

$$F' = \hat{F}'_0 \vee (\hat{F}'_1 \wedge g(H'_0, \dots, H'_\ell)).$$

Then, by induction we get that

$$\begin{aligned}
\mathrm{depth}(F') &\le 3 + \max\{\mathrm{depth}(\hat{F}'_0), \mathrm{depth}(\hat{F}'_1), \max_{i \in [\ell]} \mathrm{depth}(H'_i)\} \\
&\le 3 + \max\{3 \log \mathrm{size}(\hat{F}_0), 3 \log \mathrm{size}(\hat{F}_1), \max_{i \in [\ell]} 3 \log \mathrm{size}(H_i)\} \\
&\le 3 + 3 \log \frac{\mathrm{size}(F)}{2} = 3 \log \mathrm{size}(F).
\end{aligned}$$

□

Using Lemma 6.1, we prove Theorem 1.3.

Proof of Theorem 1.3. Suppose that the function $f : \{0,1\}^n \to \{0,1\}$ can be computed by a FOS F of size $2^{o(n/\log n)}$ over slice gates of weight bounded by $\mathrm{poly}(n)$. By Lemma 6.1, f can be computed by a FOS F' of depth $D' = o(n/\log n)$ and size $S' = 2^{o(n/\log n)}$ over slice gates of weight bounded by $\mathrm{poly}(n)$. It is shown in [2] that any (k, ℓ)-slice function can be realized with information ratio of $I = k^2$ for long secrets, i.e., for secrets of length $t = 2^{\Omega(n^k)}$ there is a secret-sharing scheme realizing the slice function with share size $O(k^2 t)$. We can use the construction of [10] (which uses a formula to construct a secret-sharing scheme) apply Lemma B.1 to F' and derive a secret-sharing scheme whose total information ratio is $O(S' I^{D'}) = 2^{o(n/\log n)} \mathrm{poly}(n)^{o(n/\log n)} = 2^{o(n)}$, as required. [16] □

[16]We note that a similar result holds if F employs k-partite gates (as per Definition 5.1) over arbitrary fan-in and with an arbitrary value of k, since such gates can be realized with constant information ratio [1]. In fact, in this case one can even start with a formula of size $2^{o(n)}$, balance it to a formula of depth $o(n)$ and size $2^{o(n)}$, and get a secret-sharing scheme whose total share size is $2^{o(n)} O(1)^{o(n)} = 2^{o(n)}$.

Acknowledgment

We thank Klim Efremenko for discussions that started this project.

The first, third and forth authors (BA, ON, NP) are supported by the European Union's Horizon 2020 Programme (ERC-StG-2014-2020) under Grant Agreement No. 639813 ERC-CLC, and by the Israel Science Foundation grant no. 2805/21. The second author (AB) is supported by ERC grant 742754 (project NTSC), Israel Science Foundation grant no. 391/21, and a grant from the Cyber Security Research Center at Ben-Gurion University.

References

[1] 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 S. Dziembowski, editors, *TCC 2018*, volume 11239 of *LNCS*, pages 317–344. Springer-Verlag, 2018.

[2] 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, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 441–471. Springer, 2019.

[3] Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 280–293. ACM, 2020.

[4] Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1.5^{n}. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 627–655. Springer, 2021.

[5] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Comb.*, 19(3):301–319, 1999.

[6] 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 *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.

[7] Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020*, volume 12552 of *Lecture Notes in Computer Science*, pages 499–529. Springer, 2020.

[8] Amos Beimel, Hussien Othman, and Naty Peter. Quadratic secret sharing and conditional disclosure of secrets. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 748–778. Springer, 2021.

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

[10] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO '88, 8th Annual International Cryptology Conference*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.

[11] Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques*, volume 718 of *Lecture Notes in Computer Science*, pages 67–79. Springer, 1992.

[12] George R. Blakley. Safeguarding cryptographic keys. In Richard E. Merwin, Jacqueline T. Zanca, and Merlin Smith, editors, *Proceedings of the 1979 AFIPS National Computer Conference*, volume 48 of *AFIPS Conference proceedings*, pages 313–317. AFIPS Press, 1979.

[13] Maria Luisa Bonet and Samuel R. Buss. Size-depth tradeoffs for Boolean fomulae. *Inf. Process. Lett.*, 49(3):151–155, 1994.

[14] Richard P. Brent. The parallel evaluation of general arithmetic expressions. *J. ACM*, 21(2):201–206, 1974.

[15] Nader H. Bshouty, Richard Cleve, and Wayne Eberly. Size-depth tradeoffs for algebraic formulas. *SIAM J. Comput.*, 24(4):682–705, 1995.

[16] Arkadev Chattopadhyay, Shachar Lovett, and Marc Vinyals. Equality alone does not simulate randomness. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019*, volume 137 of *LIPIcs*, pages 14:1–14:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[17] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM, 1988.

[18] László Csirmaz. The size of a share must be large. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques*, volume 950 of *Lecture Notes in Computer Science*, pages 13–22. Springer, 1994.

[19] László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Sci. Math. Hungar.*, 32(3–4):429–437, 1996.

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

[21] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures (extended abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91, 11th Annual International Cryptology Conference*, volume 576 of *Lecture Notes in Computer Science*, pages 457–469. Springer, 1991.

[22] Xudong Fu. Lower bounds on sizes of cutting plane proofs for modular coloring principles. In Paul Beam and Samuel R. Buss, editors, *Proof Complexity and Feasible Arithmetics, Proceedings of a DIMACS Workshop*, volume 39 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 135–148. DIMACS/AMS, 1996.

[23] Anna Gál and Jing-Tang Jang. A generalization of spira's theorem and circuits with small segregators or separators. *Inf. Comput.*, 251:252–262, 2016.

[24] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. *Theory Comput.*, 16:1–30, 2020.

[25] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000.

[26] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018.

[27] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006*, pages 89–98. ACM, 2006.

[28] Armin Haken and Stephen A. Cook. An exponential lower bound for the size of monotone real circuits. *J. Comput. Syst. Sci.*, 58(2):326–335, 1999.

[29] Pavel Hrubeš and Pavel Pudlák. A note on monotone real circuits. *Inf. Process. Lett.*, 131:15–19, 2018.

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

[31] Stasys Jukna. Combinatorics of monotone computations. *Comb.*, 19(1):65–85, 1999.

[32] Stasys Jukna. *Boolean Function Complexity – Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer-Verlag, 2012.

[33] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 539–550. ACM, 1988.

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

[35] Aleksej Dmitrievich Korshunov. On the number of monotone boolean menge. *Problemy kibernetiki*, 38:5–109, 1981.

[36] Jan Krajícek. Interpolation by a game. *Math. Log. Q.*, 44:450–458, 1998.

[37] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996.

[38] Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 699–708. ACM, 2018.

[39] Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018 – 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 567–596. Springer, 2018.

[40] Moni Naor and Avishai Wool. Access control and signatures via quorum secret sharing. In Li Gong and Jacques Stearn, editors, *CCS '96, Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, March 14-16, 1996*, pages 157–168. ACM, 1996.

[41] Noam Nisan. The communication complexity of threshold gates. In *Combinatorics, Paul Erdos is Eighty, in Bolyai Society Mathematical Studies*, pages 301–315, 1993.

[42] Toniann Pitassi and Robert Robere. Lifting nullstellensatz to monotone span programs over any field. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 1207–1219. ACM, 2018.

[43] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997.

[44] Alexander A. Razborov. A lower bound on the monotone network complexity of the logical permanent. *Math. Notes of the Acad. of Sci. of the USSR*, 37(6):485–493, 1985.

[45] John Riordan and Claude E. Shannon. The number of two-terminal series-parallel networks. *J. Math. Phys.*, 21(1-4):83–93, 1942.

[46] Arnold Rosenbloom. Monotone real circuits are more powerful than monotone Boolean circuits. *Inf. Process. Lett.*, 61(3):161–164, 1997.

[47] John E. Savage. *The Complexity of Computing*. John Wiley & Sons Inc., 1976.

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

[49] Bhavani Shankar, Kannan Srinathan, and C. Pandu Rangan. Alternative protocols for generalized oblivious transfer. In Shrisha Rao, Mainak Chatterjee, Prasad Jayanti, C. Siva Ram Murthy, and Sanjoy Kumar Saha, editors, *Distributed Computing and Networking, 9th International Conference, ICDCN 2008*, volume 4904 of *Lecture Notes in Computer Science*, pages 304–309. Springer, 2008.

[50] Philip M. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971.

[51] Ulfberg Stafan. *On Lower Bounds for Circuits and Selection*. Ph.D., Royal Institute of Technology, Stockholm, Sweden, 1999.

[52] Tamir Tassa. Generalized oblivious transfer by secret sharing. *Des. Codes Cryptogr.*, 58(1):11–21, 2011.

[53] Leslie G. Valiant, Sven Skyum, Stuart Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

[54] Emanuele Viola. The communication complexity of addition. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 632–651. SIAM, 2013.

[55] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011.

[56] Ingo Wegener. Relating monotone formula size and monotone depth of Boolean functions. *Inf. Process. Lett.*, 16(1):41–42, 1983.

A Improved MRF and FOS Sizes via Duality for Some Function Families

In this section, we present a $2^{\Omega(n)}$ gap between the best known share size in secret-sharing schemes and the sizes of FOSs and MRFs for uniformly chosen DNFs of $\Omega(n)$ width. Along the way, we prove that MRCs and FOSs are closed under duality – an interesting property that may be useful elsewhere.

To simplify the discussion in this section, we will view the inputs and outputs of Boolean functions as -1 and 1 instead of 0 and 1, where each 0 is simply replaced with -1. The *dual* of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, denoted $\mathcal{D}(f)$, is the Boolean function

$$\mathcal{D}(f)(x_1,\ldots,x_n) = -f(-x_1,\ldots,-x_n).$$

We will also extend this definition to functions $f : \mathbb{R}^n \to \mathbb{R}$. For a gate G, we denote by $\mathcal{D}(G)$ the dual gate of G.

We list a few examples of duality in the Boolean world. The dual of OR is AND and vice versa, the dual of (k,n)-threshold functions are $(n-k+1)$-threshold functions, and the dual of any (k,n)-slice function is the corresponding $(n-k,n)$-slice function. It is a long-standing open question whether the share size of a secret-sharing scheme realizing f and its dual are the same for every monotone function. See, e.g., [20]. The state of affairs today is that some functions have secret-sharing schemes with significantly smaller share sizes than known schemes for their duals. We will show that the answer to the analogues question for circuits and formulas over monotone real gates and slice gates is positive:

Claim A.1. *Let C be a circuit with gates $G_1,\ldots G_k$ that computes a function $f : \mathbb{R}^n \to \mathbb{R}$. Then, a circuit C' with the same structure and with every gate G_i replaced with $\mathcal{D}(G_i)$ computes the function $\mathcal{D}(f)$.*

Proof. We prove the claim by induction on the depth of the circuit. For the base case where the circuit has only one gate the claim is trivial. We then assume that the claim holds for circuits of depth d, and consider the root gate G of a circuit C of depth $d+1$ that computes the function f. Denote by $G_1,\ldots,G_k$ the children of G. For every $i \in [k]$, it holds that G_i is the root of a circuit C_i of depth at most d that computes some function f_i. That is, $f(x) = G(f_1(x),\ldots,f_k(x))$. By our assumption, for every $i \in [k]$, if we replace all gates in C_i with their duals, we will get a circuit C'_i computing $\mathcal{D}(f_i)$. Then, when we also replace the root G with its dual, we will get a circuit C' that computes

$$\mathcal{D}(G)[\mathcal{D}(f_1)(x),\ldots,\mathcal{D}(f_k)(x)] = -G[-(-f_1(-x)),\ldots,-(-f_k(-x))] = -G[f_1(-x),\ldots,f_k(-x)],$$

which equals $\mathcal{D}(f)$ as desired. □

Lemma A.2 (Duality for circuits with real gates and slice gates)**.** *If a Boolean function f has a circuit C with slice gates and monotone real gates of size s, then the dual of f, $\mathcal{D}(f)$, has a circuit C' with the same structure and size s (but with different specifications for the slice gates and the real gates). Furthermore, if C contains only slice gates, then C' has only slice gates, and if C contains only real gates, then C' has only real gates.*

Proof. Note that when G is a monotone real gate then $\mathcal{D}(G)$ also computes a monotone real function: If $x < y$, then $-y < -x$, and since G is monotone, $\mathcal{D}(G)(x) = -G(-x) \leq -G(-y) = \mathcal{D}(G)(y)$. As mentioned before, it is also clear that when G is a slice gate then $\mathcal{D}(G)$ computes a slice function (with a different slice parameter). Thus, Claim A.1 implies the lemma. □

We next discuss an application of Lemma A.2.

Definition A.3 (The (a, k, n)-DNF distribution [4])**.** *For positive integers n, $a \leq n$, and $1 \leq k \leq \binom{n}{a}$, we define the (a, k, n)-*DNF distribution *over monotone functions $f : \{0,1\}^n \to \{0,1\}$ as follows: Sample k distinct n-bit strings $y_1, \dots, y_k$ of Hamming weight a, and take f to be the monotone function whose minterms are $y_1, \dots, y_k$.*

Applebaum and Nir [4] showed that if share sizes were the same for monotone functions and their duals, better secret-sharing schemes could be realized for the above distribution of functions (with high probability over the choice of the function). Similar to the other constructions discussed in this paper (see Section 5), their construction first implicitly constructs constant depth formulas over slice gates for some functions. Then they assume that the duals of these functions have secret-sharing schemes with the same share size, and under this assumption constructs better secret-sharing schemes for functions sampled from the (a, k, n)-DNF distribution. By Lemma A.2, the constant depth formulas over slice gates have corresponding constant depth formulas over slice gates for the dual functions. Thus, plugging these constant depth FOSs for the dual functions in the constant depth formula of [4] over slice gates results in a constant depth FOS for functions from (a, b, n)-DNF distribution of size $O(2^{0.491n})$ for every values of $a = a(n)$ and $b = b(n)$. In addition, these FOSs can be translated to MRFs using the construction of Rosenbloom [46], obtaining a constant depth formula over real gates for functions from (a, b, n)-DNF distribution of size $O(2^{0.491n})$.[17]

To conclude, for FOSs and MRFs we get the following theorem, which is the FOS or MRF version of [4, Theorem 6.2]. While [4, Theorem 6.2] contains an assumption on secret-sharing schemes (which might or might not hold),[18] the statement in the next theorem has no assumptions.

Theorem A.4. *For every functions $a = a(n), b = b(n)$, a function sampled from the (a, b, n)-DNF distribution has a FOS and an MRF of size at most $2^{0.491n+o(n)}$ with probability $1 - 2^{-\Omega(n)}$.*[19]

In contrast, the best known secret-sharing upper-bound for the (a, b, n)-DNF distribution (for arbitrary a, b) is $2^{0.5n+o(n)}$.

[17]Alternatively, we can translate the construction of [4] using the result of [46] and apply Lemma A.2 for formulas over real gates.

[18]The slice functions used in all known secret-sharing constructions are very sparse, i.e., (k, n)-slices where $k << n$; it is not known how to realize their dual “dense” slices, where k is close to n, with similar share sizes. Moreover, it is not clear if such construction exists.

[19]The value 0.491 is the one for which the following equation holds: $\frac{1}{2}\,\mathrm{H}_2(\lambda) - (1-\lambda)\,\mathrm{H}_2(\frac{\lambda}{1-\lambda}) = 0$.

B From Generalized Boolean Formulas to Secret Sharing

It is well known that monotone Boolean formulas (over the standard AND, OR basis) give rise to secret-sharing schemes of similar complexity [10]. The following (folklore) lemma generalizes this relation to monotone formulas over a general basis and can be proved by induction on the number of gates.

Lemma B.1. *Suppose that a function $f : \{0,1\}^n \to \{0,1\}$ 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 maximal share-size is w_g. Then, f can be realized by a secret-sharing scheme whose total share size is w_F where the weight function w_F is defined as follows.*

- *The weight $w_F(v)$ of a leaf v in F is the product $\prod_i w_{g_i}$ where g_i is the ith gate in the (unique) path from the root to v.*
- *The weight w_F of the formula F is the total weight of all its leaves.*

Similarly, if every gate g can be realized by a secret-sharing scheme whose maximal information ratio is w_g, then f can be realized with total information ratio of w_F.

Note that any non-trivial gate $g \neq 0$ has weight of at least 1, and so the formula size (measured as the number of leaves) forms a lower-bound on w_F. Indeed, when the basis of F solely consists of OR and AND gates whose weight is exactly 1, the lower-bound is tight.

Sketch of proof of Lemma B.1. The proof of the lemma is by induction on the depth of the formula. For a depth 0 formula, i.e., a single-gate formula F with gate g, it is straightforward that f can be realized by a secret-sharing scheme with total share size $w_F = w_g$.

For a formula F with depth at least 1 whose root gate is g, let $g_1, \dots, g_\ell$ be the children of g, and for every $i \in [\ell]$, let F_i be the sub-formula of F rooted at g_i and f_i be the function computed by F_i. By induction, f_i can be realized by a secret-sharing scheme with total share size w_{F_i}. Then, the function f can be realized by a secret-sharing scheme in which we first share the secret s using the scheme for g to get shares $s_1, \dots, s_\ell$, each of size w_g (by the definition of w_g), and then share each bit of s_i using the secret-sharing scheme for f_i with total share size w_{F_i} promised by the induction hypothesis.

The correctness and privacy of the resulting scheme follows via a standard argument. Let us analyze its complexity. We observe that for a leaf v in F_i, the weight of v in F is the product of all the weights of the gates in the path from the root g to v (by the first item of the definition of w_F), which is the product of all the weights of the gates in the path from the root g_i of F_i to v (which is the weight of v in F_i) multiplied by the weight of g, i.e., $w_F(v) = w_g w_{F_i}(v)$. Also, the weight of F is the total weights of its leaves (by the second item of the definition of w_F), i.e., $w_F = \sum_{v \in F} w_F(v)$ and $w_{F_i} = \sum_{v \in F_i} w_{F_i}(v)$ for every $i \in [\ell]$.

The resulting scheme realizes $f = g(f_1, \dots, f_\ell)$ with total share size is

$$\sum_{i=1}^{\ell} w_g w_{F_i} = \sum_{i=1}^{\ell} w_g \sum_{v \in F_i} w_{F_i}(v) = \sum_{i=1}^{\ell} \sum_{v \in F_i} w_g w_{F_i}(v) = \sum_{i=1}^{\ell} \sum_{v \in F_i} w_F(v) = \sum_{v \in F} w_F(v) = w_F.$$

This completes the proof of Lemma B.1. □