The Power of Negations in Cryptography*

Siyao Guo[1], Tal Malkin[2], Igor C. Oliveira[2], and Alon Rosen[3]

[1] Department of Computer Science and Engineering, Chinese Univ. of Hong Kong, syguo@cse.cuhk.edu.hk

[2] Department of Computer Science, Columbia University, {tal,oliveira}@cs.columbia.edu

[3] Efi Arazi School of Computer Science, IDC Herzliya, Israel, alon.rosen@idc.ac.il

Abstract. The study of monotonicity and negation complexity for Boolean functions has been prevalent in complexity theory as well as in computational learning theory, but little attention has been given to it in the cryptographic context. Recently, Goldreich and Izsak (2012) have initiated a study of whether cryptographic primitives can be monotone, and showed that one-way functions can be monotone (assuming they exist), but a pseudorandom generator cannot.

In this paper, we start by filling in the picture and proving that many other basic cryptographic primitives cannot be monotone. We then initiate a *quantitative* study of the power of negations, asking how many negations are required. We provide several lower bounds, some of them tight, for various cryptographic primitives and building blocks including one-way permutations, pseudorandom functions, small-bias generators, hard-core predicates, error-correcting codes, and randomness extractors. Among our results, we highlight the following.

- Unlike one-way functions, one-way permutations cannot be monotone.
- We prove that pseudorandom functions require $\log n - O(1)$ negations (which is optimal up to the additive term).
- We prove that error-correcting codes with optimal distance parameters require $\log n - O(1)$ negations (again, optimal up to the additive term).
- We prove a general result for monotone functions, showing a lower bound on the depth of any circuit with t negations on the bottom that computes a monotone function f in terms of the monotone circuit depth of f. This result addresses a question posed by Koroth and Sarma (2014) in the context of the circuit complexity of the Clique problem.

* The first author was partially supported by RGC GRF grant CUHK 410111. The second and the third author were supported in part by NSF grants CCF-116702 and CCF-1423306. The first and the third author did part of this work while visiting IDC Herzliya, supported by the ERC under the European Union's Seventh Framework Programme (FP/2007-2013) Grant Agreement n. 307952. The fourth author was supported by ISF grant no. 1255/12 and by the ERC under the European Union's Seventh Framework Programme (FP/2007-2013) Grant Agreement n. 307952.

1 Introduction

Why do block ciphers like AES (Advanced Encryption Standard) have so many XOR gates dispersed throughout the levels of its circuit? Can we build a universal hard-core bit alternative to the Goldreich and Levin one [20] that only applies a small (say, constant) number of XORs? Why does the Goldreich, Goldwasser, and Micali [18] construction of a pseudorandom function (PRF) from a pseudorandom generator (PRG) heavily rely on selection functions, and calls the PRG many times? Could there be a monotone construction of a PRF from a PRG?

These are a few of the many fascinating questions related to the negation complexity of cryptographic primitives. The *negation complexity* of a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is the minimum number of negation gates in any fan-in two circuit with AND, OR, and NOT gates computing f. Note that negation gates are equivalent to XOR gates (of fan-in 2), in the sense that any circuit with t negation gates can be transformed into an equivalent circuit with t XOR gates, and vice-versa.[4] A function is *monotone* if and only if its negation complexity is 0.

In this paper, we initiate the investigation of the negation complexity of cryptographic primitives. We take first steps in this study, providing some surprising results, as well as pointing to some basic, intriguing problems that are still open.

This direction fits within the larger program of studying how *simple* basic cryptographic primitives can be, according to various complexity measures such as required assumptions, minimal circuit size, depth, etc (see, e.g., [4]). Exploring such questions helps gaining a deeper theoretical understanding of fundamental primitives and the relationships among them, and may provide the basis for understanding and addressing practical considerations as well.

While the study of monotone classes of functions and negation complexity has been prevalent in circuit complexity ([21,3,38,36,35,7,6,29,28], to name a few) and computational learning theory (see e.g. [8,9,11,32,13]), little attention has been given to it in the cryptographic context.

Recently, Goldreich and Izsak [19] have initiated a study of "cryptography in the monotone world", asking whether basic cryptographic primitives may be monotone. They focus on one-way functions (OWF) and pseudorandom generators, and show an inherent gap between the two by proving: (1) if any OWF exist, then there exist OWFs with polynomial-size monotone circuits, but (2) no monotone function can be a PRG. Quoting from their paper: *these two results indicate that in the "monotone world" there is a fundamental gap between one-way functions and pseudorandom generators; thus, the "hardness-vs-randomness" paradigm fails in the monotone setting.* This raises the following natural question:

> *Can other cryptographic primitives be computed by polynomial-size monotone circuits?*

[4] $\neg x$ is equivalent to $x \oplus 1$, while $x \oplus y$ is equivalent to $\neg(x \wedge y) \wedge (x \vee y)$.

We consider this question for several primitives and building blocks, showing negative answers for all of them. This may suggest the interpretation (or conjecture) that in the "monotone world", there is no cryptography except for one-way functions. We then initiate a *quantitative* study (where our main contributions lie), putting forward the question:

> *How many negations are required (for poly-size circuits) to compute fundamental cryptographic building blocks?*

Markov [27] proved that the negation complexity of any function $h\colon \{0,1\}^n \to \{0,1\}^m$ is at most $\lceil \log(n+1) \rceil$, and Fischer [14] proved that this transformation can be made efficient (see Jukna [22] for a modern exposition). In light of these results, is it the case that all natural cryptographic primitives other than OWFs require $\Omega(\log n)$ negations, or are there primitives that can be computed with, say, a constant number of negations?

We state our results informally in the next section. Since our lower bounds hold for well-known primitives, we postpone their definitions to Section 3.

2 Our Results

Our contributions alongside previously known results are summarized in Figure 1, together with the main idea in each proof (the definition of these primitives can be found in Section 3). We explain and discuss some interesting aspects of these results below, deferring complete details to the body of the paper.

Primitive	Lower Bound	Upper Bound	Ref.	Proof Ideas
OWF	-	(monotone)	[19]	Embedding into middle slice
OWP	**non-monotone**	$\log n + O(1)$	here	Combinatorial and analytic proofs
PRG	non-monotone	$\log n + O(1)$	[19]	AND of one or two output bits
SBG	**non-monotone**	$\boldsymbol{\omega(1)}$	here	Extension of [19]; Parity of Tribes
WPRF	non-monotone	$(\frac{1}{2}+o(1))\log n$	[9]	Weak-learner for monotone functions
PRF	$\boldsymbol{\log n - O(1)}$	$\log n + O(1)$	here	Alternating chains in the hypercube
ECC	$\boldsymbol{\log n - O(1)}$	$\log n + O(1)$	here	Extension of [12]
HCB	$\boldsymbol{(\frac{1}{2}-o(1))\log n}$	$(\frac{1}{2}+o(1))\log n$	here	Low influence and [16]
EXT	$\boldsymbol{\Omega(\log n)}$	$\log n + O(1)$	here	Low noise-sensitivity and [10]

Fig. 1. Summary of the negation complexity of basic cryptographic primitives and building blocks. Boldface results correspond to new bounds obtained in this paper. The $\log n + O(1)$ upper bound is Markov's bound [27] for any Boolean function. Error-correcting codes (ECC) and extractors (EXT) refer to constructions with good distance and extraction parameters.

2.1 Cryptography is Non-Monotone

As mentioned above, [19] proved that if OWFs exist, then they can be monotone, while PRGs cannot. We fill in the picture by considering several other cryptographic primitives, and observing that none of them can be monotone (see Figure 1).

A result of particular interest is the lower bound showing that one-way permutations (OWP) *cannot* be monotone. We obtain this result by proving that any monotone permutation f on n variables must satisfy $f(x_1, \ldots, x_n) = \big(x_{\pi(1)}, \ldots, x_{\pi(n)}\big)$, for some permutation $\pi \colon [n] \to [n]$ (finding the permutation and inverting f can then be done by evaluating f on n inputs). This is surprising in light of the [19] construction for OWFs. In particular, our result can be seen as a separation between OWFs and OWPs in the monotone world.

We provide two proofs of our result. The first is based on analytical methods, and was inspired by an approach used by Goldreich and Izsak [19]. The second is more elementary, and relies on a self-contained combinatorial argument.

2.2 Highly Non-Monotone Primitives

We show that many central cryptographic primitives are highly non-monotone. Some of our lower bounds demonstrate necessity of $\log n - O(1)$ negations, which is tight in light of Markov's $\log n + O(1)$ upper bound [27]. For some of the primitives we give less tight $\Omega(\log n)$ lower bounds.

Pseudorandom Functions (PRF). We show that PRFs can only be computed by circuits containing at least $\log n - O(1)$ negations (which is optimal up to the additive term). We prove this by exhibiting an adversary that distinguishes any function that can be implemented with fewer negations gates from a random function. Our result actually implies that for any PRF family $\{F(w, \cdot)\}$, for almost all seeds w, $F(w, \cdot)$ can only be computed by circuits with at least $\log n - O(1)$ negations.[5]
The distinguisher we construct asks for the values of the function on a fixed chain from 0^n to 1^n and accept if the alternating number of this chain is large. We note that the distinguisher suceeds for any function that has an implementation with fewer negations than the lower bound, regardless of the specific implementation the PRF designer had in mind. This can be considered as another statistical test to run on proposed candidate PRF implementations.

Error-Correcting Codes (ECC). As shown by Buresh-Oppenheim, Kabanets and Santhanam [12], if an ECC has a monotone encoding function then one can find two codewords that are very close. This implies that there is no monotone ECC with good distance parameters.

[5] That is, if we consider the circuit computing the PRF family $F(\cdot, \cdot)$ as a single function (with the seed as one of the inputs), then this circuit must have at least logarithmically many negation gates.

We extend this result to show that, given a circuit with t negation gates computing the encoding function, we can find two codewords whose Hamming distance is $O(2^t \cdot m/n)$ (for codes going from n bits to m bits). Consequently, this gives a $\log n - O(1)$ lower bound on the negation complexity of ECC with optimal distance parameters.

Hard-core Bits (HCB). Recall that a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ is a hard-core predicate for a function $f\colon \{0,1\}^n \to \{0,1\}$ if, given $f(x)$, it is hard to compute $h(x)$. We show that general hard-core bit predicates must be highly non-monotone. More specifically, there exists a family of one-way functions f_n for which any hard-core predicate requires $\Omega(\log n)$ negations (assuming one-way functions exist).
Our result follows via the analysis of the *influence* of circuits with few negations, and a corresponding lower bound on hard-core bits due to Goldmann and Russell [16].

(Strong) Extractors (EXT). A strong extractor produces almost uniform bits from weak sources of randomness, even when the truly random seed used for extraction is revealed. We prove that any extractor function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^{100}$ that works for $(n, n^{1/2-\varepsilon})$-sources requires circuits with $\Omega(\log n)$ negations (see Section 3 for definitions).
This proof relies on the analysis of the *noise sensitivity* of circuits containing negations, together with a technique from Bogdanov and Guo [10].

2.3 Non-trivial Upper Bound for Small-Bias Generators

The above lower bounds may suggest the possibility that, with the exception of OWFs, all cryptographic building blocks require $\Omega(\log n)$ negations. We show one example of a primitive – small-bias generator (SBG) – that can be constructed with significantly fewer negations, namely, with any super-constant number of negations (for example, $\log^*(n)$ such gates).

A SBG can be thought of as a weaker version of a PRG, where the output fools linear distinguishers (i.e., it looks random to any distinguisher that can only apply a linear test). Thus, any PRG is also a SBG, but not vice-versa. We construct our SBG with few negations by outputting the input and an additional bit consisting of a parity of independent copies of the Tribes function.

Since SBGs are not quite a cryptographic primitive (these can be constructed unconditionally, and are not secure against polynomial adversaries), one may still conjecture that all "true" cryptographic primitives with the exception of OWFs require $\Omega(\log n)$ negations. We do not know whether this is the case, and it would be interesting to determine whether other primitives not covered in this paper can be monotone.

2.4 Lower Bounds for Boolean Circuits with Bottom Negations

In addition to studying specific primitives, we investigate general structural properties of circuits with negations. We prove a theorem showing that for monotone functions, the depth of any circuit with negations at the bottom (input) level only is lower bounded by the monotone depth complexity of the function minus the number of negations in the circuit. This improves a result by Koroth and Sarma [25] (who proved a multiplicative rather than additive lower bound), and answers an open problem posed by them in the context of the circuit complexity of the Clique problem. Our proof is inspired by ideas from [25], and relies on a circular application of the Karchmer-Wigderson connection between boolean circuits and communication protocols.

This result suggests that negations at the bottom layer are less powerful and easier to study. In the Appendix we describe some techniques (following results of Blais et al. [8]) that allow one to decompose arbitrary computations into monotone and non-monotone components, and provide further evidence that negations at the bottom are less powerful (see also the discussion in Section 6).

Organization. We provide the definitions for most of the primitives mentioned in this paper in Section 3. Basic results used later in our proofs are presented in Section 4, with some proofs deferred to Appendix A. Our main results appear in Section 5. Finally, Section 6 discusses some open problems motivated by our work.

3 Preliminaries and Notation

In this section, we set up notation and define relevant concepts. We refer the reader to the textbooks [22], [5], [17], and [26] for more background in circuit complexity, computational complexity, theory of cryptography, and communication complexity, respectively.

3.1 Basic Notation

We use $[n]$ to denote the set $\{1,\dots,n\}$. Given a Boolean string w, we use $|w|$ to denote its length, and $|w|_1$ to denote the number of 1's in w. Unless explicitly stated, we assume that the underlying probability distribution in our equations is the uniform distribution over the appropriate set. Further, we let $\mathcal{U}_\ell$ denote the uniform distribution over $\{0,1\}^\ell$. We use $\log x$ to denote a logarithm in base 2, and $\ln x$ to refer to the natural base.

Given strings $x, y \in \{0,1\}^n$, we write $x \preceq y$ if $x_i \leq y_i$ for every $i \in [n]$. A *chain* $\mathcal{X} = (x^1, \dots, x^t)$ is a monotone sequence of strings over $\{0,1\}^n$, i.e., $x^i \preceq x^{i+1}$ for every $i \in [1, t-1]$. We say that a chain $\mathcal{X} = (x^1, x^2, \dots, x^t)$ is k-*alternating* with respect to a function $f\colon \{0,1\}^n \to \{0,1\}$ if there exist indexes $i_0 < i_1 < \dots < i_k$ such that $f(x^{i_j}) \neq f(x^{i_{j+1}})$, for every $j \in [0, k-1]$. If this is true for every pair of consecutive elements of the chain, we say that the chain is *proper* (with respect to f). We let $a(f, \mathcal{X})$ be the size of the largest set of indexes satisfying this condition. The alternating complexity of a Boolean function f is given by $a(f) \overset{\text{def}}{=} \max_{\mathcal{X}} a(f, \mathcal{X})$, where $\mathcal{X}$ is a chain over $\{0,1\}^n$. A function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ whenever $x \preceq y$. A function $g\colon \{0,1\}^n \to \{0,1\}^m$ is monotone if every output bit of g is a monotone Boolean function. Moreover, we say that a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *anti-monotone* if f is the negation of a monotone Boolean function.

3.2 Boolean Circuits and Negation Gates

Every Boolean circuit mentioned in this paper consists of AND, OR and NOT gates, where the first two types of gates have fan-in two. Recall that a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone if and only if it is computed by a circuit with AND and OR gates only.

For convenience, the size of a circuit C will be measured by its number of AND and OR gates, and will be denoted by $\mathsf{size}(C)$. The depth of a circuit C, denoted by $\mathsf{depth}(C)$, is the largest number of AND and OR gates in any path from the output gate to an input variable. The depth of a Boolean function f is the minimum depth of a Boolean circuit computing f. Similarly, the depth of a *monotone* function f, denoted by $\mathsf{depth}^+(f)$, is the minimum depth among all *monotone* circuits computing f. We will also consider multi-output Boolean circuits that compute Boolean functions $f\colon \{0,1\}^n \to \{0,1\}^m$. We stress that whenever we say that a function of this form is computed by a circuit with t negations, it means that there exists a *single* circuit (with multiple output gates) containing at most t negations computing f.

We say that a circuit contains negation gates at the bottom layer only if any NOT gate in the circuit gets as input an input variable x_i, for some $i \in [n]$. We will also say that circuits of this form are DeMorgan circuits. Put another way, a circuit $C(x)$ of size s with t negations at the bottom layer can be written as $D(x, (x \oplus \beta))$, where D is a *monotone* circuit of size s, $\beta \in \{0,1\}^n$ with $|\beta|_1 = t$ encodes the variables that appear negated in C, and $x \oplus \beta \in \{0,1\}^n$ is the string obtained via the bit-wise XOR operation. This notation is borrowed from Koroth and Sarma [25], which refers to β as the orientation vector.

3.3 Complexity Measures for Boolean Functions

Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and an index $i \in [n]$, we use $I_i(f)$ to denote the *influence* of the i-th input variable on f, i.e., $I_i(f) \overset{\text{def}}{=} \Pr_x[f(x) \neq f(x^{\oplus i})]$, where $x^{\oplus i}$ denotes the string obtained from x by flipping its i-th coordinate. The *influence* of f (also known as *average-sensitivity*) is defined as $I(f) \overset{\text{def}}{=} \sum_{i\in[n]} I_i(f)$. We say that a Boolean function f is *balanced* or *unbiased* if $\Pr_x[f(x) = 1] = 1/2$. We use $\mathsf{NS}_p(f)$ to denote the *noise sensitivity* of f under noise rate $p \in [0, 1/2]$, which is defined as $\Pr[f(X \oplus Y) \neq f(X)]$, where X is distributed uniformly over $\{0,1\}^n$, and Y is the p-biased binomial distribution over $\{0,1\}^n$, i.e., each coordinate of Y is set to 1 independently with probability p.

3.4 Pseudorandom Functions and Weak Pseudorandom Functions

Let $\mathcal{F}_n$ be the set of all Boolean functions on n variables, and $F\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$. We say that F is an (s,ε)-secure *pseudorandom function* (PRF) if, for every (non-uniform) algorithm $\mathcal{A}$ that can be implemented by a circuit of size at most s,

$$\left| \Pr_{w\sim\{0,1\}^m}\left[\mathcal{A}^{F(w,\cdot)} = 1\right] - \Pr_{f\sim\mathcal{F}_n}\left[\mathcal{A}^f = 1\right] \right| \le \varepsilon,$$

where $\mathcal{A}^h$ denotes the execution of $\mathcal{A}$ with oracle access to a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ (circuits with access to oracle gates are defined in the natural way).

A *weak pseudorandom function* (WPRF) is defined similarly, except that the distinguisher only has access to *random examples* of the form $(x, F(w,x))$, where x is uniformly distributed over $\{0,1\}^n$. In particular, any (s,ε)-secure pseudorandom function is an (s,ε)-secure weak pseudorandom function, while the other direction is not necessarily true.

3.5 Pseudorandom Generators and Small-Bias Generators

A function $G\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *pseudorandom generator* (PRG) with *stretch* $\ell \overset{\text{def}}{=} m-n$ if for every circuit $C(z_1,\ldots,z_m)$ of size s,

$$\left| \Pr_{x\sim\mathcal{U}_n}[C(G(x))=1] - \Pr_{y\sim\mathcal{U}_m}[C(y)=1] \right| \le \varepsilon.$$

We say that a function $g\colon \{0,1\}^n \to \{0,1\}^m$ is an ε-secure *small-bias generator* (SBG) with stretch $\ell = m-n$ if, for every nonempty set $S \subseteq [m]$,

$$\left| \Pr_{x\sim\mathcal{U}_n,\, y=g(x)}\left[\sum_{i\in S} y_i \equiv 1 \pmod 2\right] - \frac{1}{2} \right| \le \varepsilon.$$

Observe that small-bias generators can be seen as weaker pseudorandom generators that are required to be secure against linear distinguishers only. We refer the reader to Naor and Naor [30] for more information about the constructions and applications of such generators.

3.6 One-Way Functions, One-Way Permutations, and Hard-Core Bits

We say that a function $f\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *one-way function* (OWF) if for every circuit C of size at most s,

$$\Pr_{x\sim\mathcal{U}_n,\, y=f(x)}[C(y) \in f^{-1}(y)] \le \varepsilon.$$

If $m=n$, we say that f is *length-preserving*. If in addition f is a one-to-one mapping, we say that f is an (s,ε)-secure *one-way permutation* (OWP).

We say that a function $h\colon \{0,1\}^n \to \{0,1\}$ is an (s,ε)-secure hard-core bit for a function $f\colon \{0,1\}^n \to \{0,1\}^m$ if, for every circuit C of size s,

$$\Big| \Pr_{x \sim \mathcal{U}_n} [C(f(x)) = h(x)] - \frac{1}{2} \Big| \leq \varepsilon.$$

3.7 Extractors and Error-Correcting Codes

The *min-entropy* of a random variable X, denoted by $H_\infty(X)$, is the largest real number k such that $\Pr[X = x] \leq 2^{-k}$ for every x in the range of X. A distribution X over $\{0,1\}^n$ with $H_\infty(X) \geq k$ is said to be an (n,k)-source. Given random variables X and Y with range $\{0,1\}^m$, we let

$$\delta(X,Y) \overset{\text{def}}{=} \max_{S \subseteq \{0,1\}^m} \big| \Pr[X \in S] - \Pr[Y \in S] \big|$$

denote their *statistical distance*. We say that X and Y are ε-close if $\delta(X,Y) \leq \varepsilon$.

We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is a (strong) (k,ε)-*extractor* (EXT) if, for any (n,k)-source X, the distributions $(\mathcal{U}_s, \mathsf{Ext}(X, \mathcal{U}_s))$ and $\mathcal{U}_{s+m}$ are ε-close.[6]

Given strings $y^1, y^2 \in \{0,1\}^m$, we let

$$\varDelta(y^1,y^2) \overset{\text{def}}{=} \frac{|\{i \in [m] \mid y^1_i \neq y^2_i\}|}{m}$$

be their *relative Hamming distance*. Given a function $E\colon \{0,1\}^n \to \{0,1\}^m$, we say that E has *relative distance* γ if for every distinct pair of inputs $x^1, x^2 \in \{0,1\}^n$, we have $\varDelta(E(x^1), E(x^2)) \geq \gamma$. As a convention, we will refer to a function of this form as an *error-correcting code* (ECC) whenever we are interested in the distance between its output strings (also known as "codewords").

4 Basic Results and Technical Background

4.1 Karchmer-Wigderson Communication Games

Karchmer-Wigderson games [24] are a powerful tool in the study of circuit depth complexity. We focus here on games for *monotone* functions. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a monotone function, and consider the following deterministic communication game between two players named Alice and Bob. Alice gets an input $x \in f^{-1}(1)$, while Bob receives $y \in f^{-1}(0)$. The goal of the players is to communicate the minimum number of bits (using an interactive protocol), and to output a coordinate $i \in [n]$ for which $x_i = 1$ and $y_i = 0$. The monotonicity assumption on f guarantees that such coordinate always exists.

Proposition 1 (Karchmer and Wigderson [24]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a monotone function. There exists a monotone circuit C_f of depth d that computes f if, and only if, there exists a protocol $\varPi_f$ for the* (*monotone*) KW-*game of f with communication cost d.*

[6] Two occurrences of the same random variable in an expression refer to the same copy of the variable.

4.2 Markov's Upper Bound

The following result was obtained by Markov [27].

Proposition 2 (Markov [27]). *Let $f\colon \{0,1\}^n \to \{0,1\}^m$ be an arbitrary function. Then f is computed by a (multi-output) Boolean circuit containing at most $\lceil \log(n+1) \rceil$ negations.*

This result implies that many of our lower bounds are tight up to an additive term independent of n. Some of our proofs also rely on the following relation between negation complexity and alternation.

Proposition 3 (Markov [27]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit with at most t negations. Then $a(f) = O(2^t)$.*

4.3 The Flow of Negation Gates

It is useful in some situations to decompose the computation of a function into monotone and non-monotone components. This idea has been applied successfully to obtain almost optimal bounds on the learnability of functions computed with few negation gates (Blais et al. [8]). An important lemma used in their paper can be stated as follows.

Lemma 1 (Blais et al. [8]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C with at most t negations. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is monotone, $T = O(2^t)$, and h is either the parity function, or its negation.*

A drawback of this statement is that the computational complexity of each g_i is not related to the size of C. Roughly speaking, the proof of this result uses a circuit for f in order to gain structural information about f, and then rely on a non-constructive argument. We observe that, by relaxing the assumption on h, we can prove the following effective version of Lemma 1.[7]

Lemma 2. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s containing t negation gates. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is computed by a monotone circuit of size at most s, $T = 2^{t+1} - 1$, and $h\colon \{0,1\}^T \to \{0,1\}$ is computed by a circuit of size at most $5T$.*

We state below a more explicit version of Lemma 2 for circuits with a single negation gate and several output bits. The proof of this result follows from the same argument used to derive Lemma 2, whose proof we give in Section A.

[7] This result was obtained during a discussion with Clement Canonne, Li-Yang Tan, and Rocco Servedio.

Lemma 3. *Let $f\colon \{0,1\}^n \to \{0,1\}^u$ be computed by a circuit of size s containing a single negation gate. Assume that the j-th output bit of f is computed by the function $f_j\colon \{0,1\}^n \to \{0,1\}$. Then, there exist monotone functions $m\colon \{0,1\}^n \to \{0,1\}$ and $m_{j,\ell}\colon \{0,1\}^n \to \{0,1\}$, where $j \in [u]$ and $\ell \in \{0,1\}$, which are computed by monotone circuits of size at most s, and a function $h\colon \{0,1\}^3 \to \{0,1\}$, such that:*

(i) For every $j \in [u]$, $f_j(x) = h(m(x), m_{j,0}(x), m_{j,1}(x))$.
(ii) For every $j \in [u]$ and $x \in \{0,1\}^n$, $m_{j,0}(x) \leq m_{j,1}(x)$.
(iii) The function h is defined as $h(z, y_1, y_0) \overset{\text{def}}{=} y_z$.

From a programming perspective, Lemma 3 shows that a single negation gate in a Boolean circuit can be interpreted as an IF-THEN-ELSE statement involving monotone functions. Conversely, the selection procedure computed by h can be implemented with a single negation.

For convenience of the reader, we sketch the proof of these results in Section A, where we also discuss the expressiveness of negations at arbitrary locations compared to negations at the bottom layer of a circuit. Lemmas 1 and 2 can be used interchangeably in our proofs.

4.4 Useful Inequalities

Some of our proofs rely on the following results for Boolean functions.

Proposition 4 (Fortuin, Kasteleyn, and Ginibre [15]). *If $g\colon \{0,1\}^n \to \{0,1\}$ and $f\colon \{0,1\}^n \to \{0,1\}$ are monotone Boolean functions, then*

$$\Pr_x[f(x) = 1 \wedge g(x) = 1] \;\geq\; \Pr_x[f(x) = 1] \cdot \Pr_x[g(x) = 1].$$

The same inequality holds for anti-monotone functions. In particular, for monotone $f, g\colon \{0,1\}^n \to \{0,1\}$, following inequality holds

$$\Pr_x[f(x) = 0 \wedge g(x) = 0] \;\geq\; \Pr_x[f(x) = 0] \cdot \Pr_x[g(x) = 0].$$

A stronger version of this inequality that will be used in some of our proofs is presented below.

Proposition 5 (Talagrand [37]). *For any pair of monotone Boolean functions $f, g\colon \{0,1\}^n \to \{0,1\}$, it holds that*

$$\Pr_x[f(x) = 1 \wedge g(x) = 1] \;\geq\; \Pr_x[f(x) = 1] \cdot \Pr_x[g(x) = 1] + \psi\Big(\sum_{i \in [n]} I_i(f) \cdot I_i(g)\Big),$$

where $\psi(x) \overset{\text{def}}{=} c \cdot x / \log(e/x)$, and $c > 0$ is a fixed constant independent of n.

Proposition 6 (Kahn, Kalai, and Linial [23]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a balanced Boolean function. Then there exists an index $i \in [n]$ such that $I_i(f) = \Omega\Big(\frac{\log n}{n}\Big)$.*

Finally, we will make use of the following standard concentration bound (cf. Alon and Spencer [2]).

Proposition 7. *Let $X_1, \ldots, X_m$ be independent $\{0,1\}$ random variables, where each X_i is 1 with probability $p \in [0,1]$. In addition, set $X \overset{\text{def}}{=} \sum_i X_i$, and $\mu \overset{\text{def}}{=} \mathbb{E}[X] = pm$. Then, for any fixed $\zeta > 0$, there exists a constant $c_\zeta > 0$ such that*

$$\Pr[\,|X - \mu| > \zeta\mu] \;<\; 2e^{-c_\zeta \mu}.$$

5 Main Results

5.1 One-Way Functions versus One-Way Permutations

Goldreich and Izsak [19] proved that if one-way functions exist, then there are *monotone* one-way functions. We show below that this is not true for *one-way permutations.* In other words, one-way permutations are inherently non-monotone. This lower bound follows easily via the following structural result for monotone permutations.

Proposition 8. *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a one-to-one function. If f is monotone, then there exists a permutation $\pi\colon [n] \to [n]$ such that, for every $x \in \{0,1\}^n$, $f(x) = x_{\pi(1)} \ldots x_{\pi(n)}$. In particular, there exists a (uniform) polynomial size circuit that inverts f on every input $y = f(x)$.*

Proof. Let $f_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function corresponding to the i-th output bit of f. Since f is monotone, each function f_i is monotone. Consider functions f_ℓ and f_k, where $\ell \neq k$. By Talagrand's inequality (Proposition 5),

$$\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] \geq \Pr_x[f_\ell(x) = 1] \cdot \Pr_x[f_k(x) = 1] + \psi\Big(\sum_{i\in[n]} I_i(f_\ell) \cdot I_i(f_k)\Big). \quad (1)$$

Since f is a permutation, $\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] = 1/4$ and $\Pr_x[f_\ell(x) = 1] = \Pr_x[f_k(x) = 1] = 1/2$. Consequently, it follows from Equation 1 and the definition of ψ that

$$\sum_{i\in[n]} I_i(f_\ell) \cdot I_i(f_k) = 0.$$

In other words, f_ℓ and f_k depend on a disjoint set of input variables. Since this is true for every pair ℓ and k with $\ell \neq k$, and every output bit of f is non-constant, there exists a permutation $\pi\colon [n] \to [n]$ such that, for every $i, j \in [n]$, if $I_i(f_j) > 0$ then $i = \pi(j)$. Moreover, as f is monotone and one-to-one, we must have $f_j(x) = x_{\pi(j)}$, for every $j \in [n]$. The corresponding permutation can be easily recovered from f by evaluating this function on every indicator string $e^i \in \{0,1\}^n$, where $e^i_j = 1$ if and only if $i = j$. This completes the proof of our result.

We remark that a simple extension of this proof allows us to rule out monotone one-way functions $f\colon \{0,1\}^n \to \{0,1\}^{n-k}$ where each pre-image set $f^{-1}(x)$ has size exactly 2^k (i.e., regular OWFs), and some relaxations of this notion.

Proposition 8 implies that any circuit computing a one-way permutation contains at least one negation gate. It is not clear how to extend its proof to obtain a stronger lower bound on the negation complexity of one-way permutations, as Talagrand's inequality holds for monotone functions only. Although we leave open the problem of obtaining better lower bounds, we give next an alternative proof of Proposition 8 that does not rely on Talagrand's result.

Proof. Let $S_k \overset{\text{def}}{=} \{x \in \{0,1\}^n \mid |x|_1 = k\}$, where $k \in [0,n]$. In other words, S_k is simply the k-th slice of the n-dimensional Boolean hypercube. Initially, we prove the following claim: For every set S_k, $f(S_k) = S_k$. In other words, f induces a permutation over each set of inputs S_k. We then use this result to establish Proposition 8.

First, observe that $f(0^n) = 0^n$. Otherwise, there exists an input $x \neq 0^n$ such that $f(x) = 0^n$. Since $0^n \preceq x$ and f is monotone, we get that $f(0^n) \preceq f(x)$, which contradicts the injectivity of f. This establishes the claim for S_0. The general case follows by induction on k. Assume the result holds for any $k' < k$, and consider an arbitrary $y \in S_k$. Since f is one-to-one, there exists $x \in S_\ell$ such that $f(x) = y$, where $\ell \geq k$. If $\ell \neq k$, there exists $x' \prec x$ such that $x' \in S_k$. Let $y' \overset{\text{def}}{=} f(x')$. Using that f is monotone and $x' \prec x$, we get that $y' \preceq y$. Since f is one-to-one, $y' \prec y$, thus $y' \in S_{k'}$ for some $k' < k$. This is in contradiction with our induction hypothesis and the injectivity of f, since $f(S_{k'}) = S_{k'}$, $x' \in S_k$, and $y' = f(x') \in S_{k'}$. This completes the induction hypothesis, and the proof of our claim.

Now let $\pi\colon [n] \to [n]$ be the permutation such that $f^{-1}(e^i) = e^{\pi(i)}$, where $e^j \in \{0,1\}^n$ is the input with 1 at the j-th coordinate only. Clearly, for every $x \in S_0 \cup S_1$, $f(x) = x_{\pi(1)} \ldots, x_{\pi(n)}$. On the other hand, for every $x \in S_k$ with $k > 1$, it follows from the monotonicity of f that

$$\bigvee_{i\,:\,x_i=1} f(e^i) \;\preceq\; f(x),$$

where the disjunction is done coordinate-wise. Finally, it follows from our previous claim that we must also have $f(x) \in S_{|x|_1}$. Therefore,

$$\bigvee_{i\,:\,x_i=1} f(e^i) \;=\; f(x).$$

Consequently, for every $x \in \{0,1\}^n$, it follows that $f(x) = x_{\pi(1)} \ldots x_{\pi(n)}$, which completes the proof.

5.2 Pseudorandom Generators and Small-Bias Generators

In contrast to the situation for one-way functions, Goldreich and Izsak [19] presented an elegant proof that pseudorandom generators cannot be monotone.

More specifically, their result shows that the output distribution of a monotone function $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ can be distinguished from random either by the projection of one of its output bits, or via the conjunction of two output bits.

Recall from Section 3 that small-bias generators can be seen as restricted pseudorandom generators that are only required to be secure against linear tests. We prove next that the techniques from [19] can be used to show that there are no $(1/n^{\omega(1)})$-secure monotone small-bias generators with 1 bit of stretch. We observe later in this section that such generators can be constructed with any super-constant number of negation gates.

Proposition 9. *For any monotone function $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$, there exists a (non-uniform) linear test $D\colon \{0,1\}^{n+1} \to \{0,1\}$ such that*

$$\Big| \Pr_{x\sim\mathcal{U}_n}[D(G(x))=1] - \frac{1}{2} \Big| = \Omega\Big(\frac{1}{n^2}\Big).$$

Proof. The proof follows closely the argument in [19], combined with an appropriate application of the FKG inequality (Proposition 4). Let $G_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function corresponding to the i-th output bit of G, where $i \in [n+1]$. Observe that if there exists i such that

$$\Big| \Pr_{x\sim\mathcal{U}_n}[G_i(x)=1] - \frac{1}{2} \Big| = \Omega\Big(\frac{1}{n^2}\Big),$$

then there is a trivial linear distinguisher for G.

Assume therefore that, for every $i \in [n+1]$, G_i is almost balanced. In particular, each function G_i is $\delta(n)$-close under the uniform distribution to an unbiased function $\widetilde{G}_i\colon \{0,1\}^{n+1} \to \{0,1\}$, where $\delta(n) = o((\log n)/n)$. It follows from Proposition 6 that each function $\widetilde{G}_i$ has an influential variable. More precisely, there exists $\gamma\colon [n+1] \to [n]$ such that

$$I_{\gamma(i)}(\widetilde{G}_i) = \Omega\Big(\frac{\log n}{n}\Big),$$

for every $i \in [n+1]$. As each G_i is $\delta(n)$-close to $\widetilde{G}_i$, it follows that $I_{\gamma(i)}(G_i) = \Omega\big(\frac{\log n}{n}\big)$ as well.

By the pigeonhole principle, there exist distinct indexes i and j such that $\gamma(i) = \gamma(j)$. It follows from Proposition 5 that

$$\begin{aligned}
&\Pr_x[G_i(x)=1 \land G_j(x)=1]\\
&\geq \Pr_x[G_i(x)=1]\cdot\Pr_x[G_j(x)=1] + \psi\Big(\sum_{k\in[n]} I_k(G_i)\cdot I_k(G_k)\Big)\\
&\geq \Pr_x[G_i(x)=1]\cdot\Pr_x[G_j(x)=1] + \Omega\Big(\psi\Big(\frac{\log^2 n}{n^2}\Big)\Big)\\
&\geq \Pr_x[G_i(x)=1]\cdot\Pr_x[G_j(x)=1] + \Omega\Big(\frac{\log n}{n^2}\Big).
\end{aligned}$$

On the other hand, Proposition 4 implies that

$$\Pr_x[G_i(x) = 0 \wedge G_j(x) = 0] \;\geq\; \Pr_x[G_i(x) = 0] \cdot \Pr_x[G_j(x) = 0].$$

Combining both inequalities, and using the assumption that each output bit of G is almost balanced, we get that:

$$\begin{aligned}
&\Pr_x[G_i(x) + G_j(x) = 0] \\
&= \Pr_x[G_i(x) = 1 \wedge G_j(x) = 1] + \Pr_x[G_i(x) = 0 \wedge G_j(x) = 0] \\
&\geq \Pr_x[G_i(x) = 1] \cdot \Pr_x[G_j(x) = 1] + \Pr_x[G_i(x) = 0] \cdot \Pr_x[G_j(x) = 0] + \Omega\Big(\frac{\log n}{n^2}\Big) \\
&\geq \frac{1}{2} - O\left(\frac{1}{n^2}\right) + \Omega\Big(\frac{\log n}{n^2}\Big) \\
&= \frac{1}{2} + \Omega\Big(\frac{\log n}{n^2}\Big).
\end{aligned}$$

Therefore, the linear function $D(y) \overset{\text{def}}{=} y_i + y_j$ can distinguish the output of G from random with the desired advantage, which completes the proof.

In contrast, we show next that there *are* small-bias generators with super-polynomial security that can be computed with *any* super-constant number of negations. Let $\mathsf{Tribes}_{s,t} \colon \{0,1\}^{s \cdot t} \to \{0,1\}$ be the (monotone) Boolean function defined as

$$\mathsf{Tribes}_{s,t}(x_1, \ldots, x_{s \cdot t}) = \bigvee_{i=0}^{s-1} \bigwedge_{j=1}^{t} x_{i \cdot t + j}.$$

Further, we use $\mathsf{Tribes}_m \colon \{0,1\}^m \to \{0,1\}$ to denote the function $\mathsf{Tribes}_{s,t}$, where s is the largest integer such that $1 - (1 - 2^{-t})^s \leq 1/2$, and $t = m/s$ (i.e., we try to make Tribes as balanced as possible as a function over m variables).

Proposition 10. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be $f(x) \overset{\text{def}}{=} \oplus_{i=1}^{k} \mathsf{Tribes}_{n/k}(x^{(i)})$, where $x^{(i)}$ denotes the i-th block of x with length n/k. Let $1 \leq k(n) \leq n/\log n$, and $G \colon \{0,1\}^n \to \{0,1\}^{n+1}$ be defined by $G(x) \overset{\text{def}}{=} (x, f(x))$, Then, there exists a constant $C > 0$ such that, for any linear function $D \colon \{0,1\}^{n+1} \to \{0,1\}$,*

$$\Big|\Pr_{x \sim \mathcal{U}_n}[D(G(x)) = 1] - \frac{1}{2}\Big| \;\leq\; (C \cdot (k/n) \cdot \log(n/k))^k.$$

In particular, when $k = \omega(1)$, we can get a small-bias generator with negligible error that can be computed with roughly $\log k$ negations (via Proposition 2). Interestingly, for $k = 2$ we obtain an SBG computed with a *single negation* and security $\tilde{\Theta}(n^{-2})$, essentially matching the lower bound for *monotone* SBGs given by Proposition 9.

Proof. We assume the reader is familiar with basic concepts from analysis of Boolean functions (cf. O'Donnell [31]). Suppose that $D(y) \overset{\text{def}}{=} \sum_{i \in S} y_i \pmod 2$, where $S \subseteq [n+1]$ is nonempty. If $n+1 \notin S$, using that the first n output bits of G are uniformly distributed over $\{0,1\}^n$, we get that $|\Pr_x[D(G(x)) = 1] - 1/2| = 0$. Assume therefore that $n+1 \in S$, and let $S' \overset{\text{def}}{=} S \backslash \{n+1\}$. Then,

$$\begin{aligned} \left| \Pr_{x \sim \mathcal{U}_n} \Big[D(G(x)) = 1 \Big] - \frac{1}{2} \right| &= \left| \Pr_{x \sim \mathcal{U}_n} \Big[f(x) + \sum_{i \in S'} x_i \equiv 1 \ (\mathsf{mod}\ 2) \Big] - \frac{1}{2} \right| \\ &= \left| \Pr_{x \sim \mathcal{U}_n} \Big[\sum_{i \in S'} x_i \not\equiv f(x) \ (\mathsf{mod}\ 2) \Big] - \frac{1}{2} \right| \\ &\overset{\text{def}}{=} p. \end{aligned}$$

Let $f^- \colon \{-1,1\}^n \to \{-1,1\}$ be the corresponding version of f where we map 0 to 1, and 1 to -1, as usual. Observe that, under this correspondence,

$$\sum_{i \in S'} x_i \ \not\equiv \ f(x) \ \ (\mathsf{mod}\ 2) \quad \Longleftrightarrow \quad \chi_{S'}(x) \cdot f^-(x) \ = \ -1.$$

Therefore,

$$\begin{aligned} p &= \left| \Big(\frac{1}{2} - \frac{1}{2} \cdot \mathbb{E}_{x \sim \{-1,1\}^n} \big[\chi_{S'}(x) \cdot f^-(x) \big] \Big) - \frac{1}{2} \right| \\ &= \left| \Big(\frac{1}{2} - \frac{1}{2} \cdot \widehat{f^-}(S') \Big) - \frac{1}{2} \right| \\ &= \frac{1}{2} \cdot |\widehat{f^-}(S')|. \end{aligned}$$

In other words, in order to upper bound the distinguishing probability p, it is enough to upper bound $|\widehat{f^-}(S')|$, where $S' \subseteq [n]$. Using that $x^{(i)}$ and $x^{(j)}$ are disjoint for $i \neq j$ and $f^-(x) = \prod_{i \in [k]} \mathsf{Tribes}^-_{n/k}(x^{(i)})$, it follows that $\widehat{f^-}(S')$ is a product of Fourier coefficients of the corresponding Tribes functions. It is known that

$$\max_{T \subseteq [m]} \left| \widehat{\mathsf{Tribes}^-_m}(T) \right| = O\left(\frac{\log m}{m} \right)$$

as $m \to \infty$ (see e.g. O'Donnell [31]). Consequently, since we have $m = n/k$, we get that

$$p \ = \ \frac{1}{2} \cdot |\widehat{f^-}(S')| \ \leq \ \frac{1}{2} \cdot \max_{T \subseteq [n/k]} \left| \widehat{\mathsf{Tribes}^-_{n/k}}(T) \right|^k \ \leq \ \left(C \cdot (k/n) \cdot \log(n/k) \right)^k,$$

for an appropriate constant C.

It is possible to use other monotone functions for the construction in Proposition 10, but our analysis provides better parameters with Tribes.

5.3 Pseudorandom Functions

In this section we prove that a pseudorandon function is a highly non-monotone cryptographic primitive. For simplicity, we will not state the most general version of our result. We discuss some extensions after its proof.

Proposition 11. *If* $F\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ *is a* $(\mathsf{poly}(n), 1/3)$*-secure pseudorandom function, then any Boolean circuit computing* F *contains at least* $\log n - O(1)$ *negation gates.*

Proof. Consider the following algorithm D^h that has membership access to an arbitrary function $h\colon \{0,1\}^n \to \{0,1\}$, and computes as follows. Let $\mathcal{X} \overset{\text{def}}{=} (e^0, e^1, \ldots, e^n)$ be the chain over $\{0,1\}^n$ with $e^i \overset{\text{def}}{=} 1^i 0^{n-i}$. After querying h on each input $e^0, \ldots, e^n$ and computing $a(h, \mathcal{X})$, D accepts h if and only if $a(h, \mathcal{X}) \geq n/4$. This completes the description of D. Clearly, this algorithm can be implemented in polynomial time.

Observe that if $f \sim \mathcal{F}_n$ is a random Boolean function over n variables, then $\mathbb{E}_f[a(f, \mathcal{X})] = n/2$. In addition, it follows from a standard application of Proposition 7 that $|a(f, \mathcal{X}) - n/2| \leq n/4$ with probability exponentially close to 1. Therefore, under our assumption that F is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function,

$$\begin{aligned} \frac{1}{3} &\geq \left| \Pr_{w \sim \{0,1\}^n}[D^{F(w,\cdot)} = 1] - \Pr_{f \sim \mathcal{F}_n}[D^f = 1] \right| \\ &\geq \left| \Pr_{w \sim \{0,1\}^n}[D^{F(w,\cdot)} = 1] - (1 - o(1)) \right|, \end{aligned}$$

which implies in particular that $\Pr[D^{F(w,\cdot)} = 1] \geq 2/3 - o(1)$. Therefore, there must exist some seed w^* for which the resulting function $F_{w^*} \overset{\text{def}}{=} F(w^*, \cdot)$ over n-bit inputs satisfies $a(F_{w^*}, \mathcal{X}) \geq n/4$. It follows from Proposition 3 that if C is a circuit with t negations computing F_{w^*}, then

$$n/4 \quad \leq \quad a(F_{w^*}, \mathcal{X}) \quad \leq \quad a(F_{w^*}) \quad \leq \quad c \cdot 2^t,$$

where c is a fixed positive constant. Consequently, $t \geq \log n - O(1)$. Finally, it is clear that any circuit for F also requires $\log n - O(1)$ negations, which completes the proof.

Note that we can replace $1/3$ with any constant in $[0, 1)$. The proof of Proposition 11 also implies that if F is a sufficiently secure pseudorandom function, then for most choices of the seed $w \in \{0,1\}^m$, the resulting function $F(w, \cdot)$ over n input variables requires $\log n - O(1)$ negations. Further, observe that our distinguisher is quite simple, and makes $n + 1$ *non-adaptive* queries.

The same proof does not work for *weak* pseudorandom functions. In this case, most random examples obtained from the oracle are concentrated around the middle layer of the hypercube, and one cannot construct a chain. We remark, however, that weak pseudorandom functions cannot be monotone, as there are weak learning algorithms for the class of monotone functions (cf. Blum, Burch, and Langford [9]). We discuss the problem of obtaining better lower bounds for WPRFs in Section 6. (The upper bound on the negation complexity of WPRFs follows via standard techniques, see Section 5.5 and Blais et al. [8].)

5.4 Error-Correcting Codes

In this section, we show that circuits with few negations cannot compute error-correcting codes with good parameters. The proof generalizes the argument given by Buresh-Oppenheim, Kabanets and Santhanam [12] in the case of *monotone* error-correcting codes.

Proposition 12. *Let $E\colon \{0,1\}^n \to \{0,1\}^m$ be an error-correcting code with relative distance $\gamma > 0$. If C is a circuit with t negations that computes E, then $t \geq \log n - \log(1/\gamma) - 1$.*

Proof. Assume that $E\colon \{0,1\}^n \to \{0,1\}^m$ is computed by a (multi-output) circuit C^0 with t negation gates, and let $x_1, \ldots, x_n$ be its input variables. For convenience, we write C^0_i to denote the Boolean function computed by the i-th output gate of C^0. We proceed as in the proof of Lemma 2. More precisely, we remove one negation gate during each step, but here we also inspect the behavior of the error-correcting code on a particular set of inputs of interest. Let $\mathcal{X} \overset{\text{def}}{=} (e^0, e^1, \ldots, e^n)$ be the chain over $\{0,1\}^n$ with $e^i \overset{\text{def}}{=} 1^i 0^{n-i}$.

It follows from an easy generalization of Lemma 3 that there exist functions $f\colon \{0,1\}^n \to \{0,1\}$, $h\colon \{0,1\}^3 \to \{0,1\}$, and $g_{i,b}\colon \{0,1\}^n \to \{0,1\}$, where $i \in [m]$ and $b \in \{0,1\}$, for which the following holds.

- f is monotone;
- h is the addressing function $h(a, d_0, d_1) \overset{\text{def}}{=} d_a$;
- for every $x \in \{0,1\}^n$ and $i \in [m]$,

$$E(x)_i = h(f(x), g_{i,0}(x), g_{i,1}(x)).$$

- there exist (multi-output) circuits $C^{1,0}$ and $C^{1,1}$ over input variables $x_1, \ldots, x_n$ such that, for every $i \in [m]$ and $b \in \{0,1\}$,

$$C^{1,b}(x)_i = g_{i,b}(x).$$

- each circuit $C^{1,b}$ contains at most $t-1$ negations.

Since $e^0 \prec e^1 \prec \ldots \prec e^n$ and f is monotone, there exists $k \in [0,n]$ such that $f(e^\ell) = 0$ if and only if $\ell < k$. By the pigeonhole principle, f is constant on a (continuous) subchain $\mathcal{X}^1 \subseteq \mathcal{X}$ of size at least $(n+1)/2$, and there exists a constant $b \in \{0,1\}$ such that

$$E(e_i) = g_{1,b}(e^i) \ldots g_{m,b}(e^i),$$

whenever $e^i \in \mathcal{X}^1$. Consequently, there exists a (multi-output) circuit C^1 computed with at most $t-1$ negations that agrees with E on every $e^i \in \mathcal{X}^1$.

Observe that this argument can be applied once again with respect to $\mathcal{X}^1$ and C^1. Therefore, it is not hard to see that there must exist a chain $\mathcal{X}^t \subseteq \mathcal{X}$ of size $w \geq (n+1)/2^t$ and a *monotone* (multi-output) circuit C^t such that

$$C^t(e^i) = E(e^i),$$

for every $e^i \in \mathcal{X}^t$.

Assume that $\mathcal{X}^t = (e^j, e^{j+1}, \ldots, e^{j+w-1})$, and let $\mathcal{Y} \overset{\text{def}}{=} (y^j, \ldots, y^{j+w-1})$, where $y^i \overset{\text{def}}{=} E(e^i)$. Since C^t is *monotone* and $\mathcal{X}^t$ is a chain over $\{0,1\}^n$, we get that $\mathcal{Y}$ is a chain over $\{0,1\}^m$. By the pigeonhole principle, there exists an index $k \in [j+1, j+w-1]$ for which $y^{j-1} \preceq y^j$ and $|y^j|_1 - |y^{j-1}|_1 \leq (m+1)/w$. Now using that E computes an error-correcting code of relative distance at least γ, it follows that

$$\gamma \quad \leq \quad \Delta(y^j, y^{j-1}) \quad \leq \quad \frac{m+1}{w} \cdot \frac{1}{m} \quad \leq \quad \frac{2^t}{n+1} \cdot \frac{m+1}{m},$$

which completes the proof of our result.

It is possible to show via a simple probabilistic construction that there is a sequence of error-correcting codes $E_n \colon \{0,1\}^n \to \{0,1\}^{O(n)}$ with relative distance, say, $\gamma = 0.01$ (see e.g. Arora and Barak [5]). Proposition 12 implies that computing such codes requires at least $\log n - O(1)$ negation gates, which is optimal up to the additive term via Markov's upper bound (Proposition 2).

5.5 Hard-core Bits

We prove in this section that general hard-core predicates must be highly non-monotone. This result follows from a lower bound on the average-sensitivity of such functions due to Goldmann and Russell [16], together with structural results about monotone Boolean functions and Lemma 1. Roughly speaking, our result says that there are one-way functions that do not admit hardcore predicates computed with less than $(1/2) \cdot \log n$ negations (assuming that one-way functions exist).

Proposition 13. *Assume there exists a family $f = \{f_n\}_{n \in \mathbb{N}}$ of $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions, where each $f_n \colon \{0,1\}^n \to \{0,1\}^n$. Then, for every $\varepsilon > 0$, there exists a family $g^\varepsilon = \{g_n\}_{n \in \mathbb{N}}$ of (length-preserving) $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions for which the following holds. If $h = \{h_n\}_{n \in \mathbb{N}}$ is a $(\mathsf{poly}(n), n^{-\omega(1)})$-secure hard-core bit for g^ε, then for every n sufficiently large, any Boolean circuit computing h_n contains at least $(1/2 - \varepsilon)\log n$ negations.*

Proof. It follows from the main result of Goldmann and Russell [16] that under the existence of one-way functions, there exists a one-way function family $g^\delta = \{g_n\}_{n \in \mathbb{N}}$ that only admits hard-core bit predicates with average-sensitivity $\Omega(n^{1-\delta})$. Our result follows easily once we observe that the average-sensitivity of Boolean functions computed with t negations is $O(2^t \cdot \sqrt{n})$.[8]

First, if $f \colon \{0,1\}^n \to \{0,1\}$ is a monotone Boolean function, then $I(f) = O(\sqrt{n})$ (see e.g. O'Donnell [31]). On the other hand, it follows from Lemma 1 that any Boolean function $h \colon \{0,1\}^n \to \{0,1\}$ computed by a circuit with t

[8] This result is from Blais et al. [8], and we include its short argument here for completeness.

negation gates can be written as $h(x) = P(m_1(x), \ldots, m_T(x))$, where $T = O(2^t)$, each function m_i is monotone, and P is either the parity function or its negation. Therefore, using the definition of influence,

$$I(h) \;=\; I(P(m_1, \ldots, m_T)) \;\leq\; \sum_{i \in [T]} I(m_i) \;\leq\; T \cdot O(\sqrt{n}) \;=\; O(2^t \cdot \sqrt{n}),$$

which completes the proof.

This result is almost optimal, as any function $f\colon \{0,1\}^n \to \{0,1\}$ can be $(1/n^{\omega(1)})$-approximated by a Boolean function computed with $(1/2 + o(1)) \log n$ negations (check Blais et al. [8] for more details). More precisely, if h is a hard-core bit for f, its approximator $\widetilde{h}$ is also hard-core for f, as the inputs $f(x)$ given to the distinguisher are produced with $x \sim \mathcal{U}_n$.

5.6 Randomness Extractors

In this section, we show in Proposition 14 that strong $(n^{0.5-\varepsilon}, 1/2)$-extractors can only be computed by circuits with $\Omega(\log n)$ negation gates, for any constant $0 < \varepsilon \leq 1/2$. We proceed as follows. First, we argue that such extractors must have high noise sensitivity. The proof of this result employs a technique from Bogdanov and Guo [10]. We then upper bound the noise sensitivity of circuits with few negations. Together, these claims provide a trade-off between the parameters of the extractor, and the minimum number of negations in any circuit computing the extractor.

For convenience, we view the extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ as a family of functions

$$\mathcal{H}_{\mathsf{Ext}} \stackrel{\text{def}}{=} \{h_w\colon \{0,1\}^n \to \{0,1\}^m \mid h_w = \mathsf{Ext}(\cdot, w), \text{ where } w \in \{0,1\}^s\},$$

i.e., the family of functions obtained from the extractor by fixing its seed. Similarly, every such family can be viewed as a strong extractor in the natural way.

Lemma 4. *Let $0 \leq p \leq 1/2$, $0 \leq \gamma \leq 1/4$, and $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. Assume that $\mathsf{NS}_p(h_i) \leq \gamma$ for every function $h_i\colon \{0,1\}^n \to \{0,1\}$ that computes the i-th output bit of some function in $\mathcal{H}$, where $i \in [m]$. Then there exists a distribution D over $\{0,1\}^n$ with min-entropy $H_\infty(D) = n \cdot \log(\frac{1}{1-p})$ such that the statistical distance between $(\mathcal{H}, \mathcal{H}(D))$ and $(\mathcal{H}, \mathcal{U}_m)$ is at least $(1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma})$.*

Proof. For a fixed $y \in \{0,1\}^n$, let D_y denote a random variable distributed according to $y \oplus X$, where X is the p-biased binomial distribution over $\{0,1\}^n$. Since $p \leq 1/2$, observe that the min-entropy of D_y is precisely

$$\begin{aligned} H_\infty(D_y) \;&=\; -\log \max_{z \in \{0,1\}^n} \Pr[y \oplus X = z] \;=\; -\log\, \Pr[y \,\oplus\, X = y] \\ &= -\log\,(1-p)^n = n \cdot \log\left(\frac{1}{1-p}\right). \end{aligned}$$

We will need the following result.

Claim. For any fixed $h \in \mathcal{H}$,

$$\mathrm{E}_{y\sim\{0,1\}^n}[\delta(h(D_y),\mathcal{U}_m)] \;\geq\; (1-2\sqrt{\gamma}-2^{-0.1m})(1-2\sqrt{\gamma}). \tag{2}$$

We use this claim to complete the demonstration of Lemma 4, then return to its proof. Observe that, for any fixed $y \in \{0,1\}^n$,

$$\delta((\mathcal{H},\mathcal{H}(D_y)),(\mathcal{H},\mathcal{U}_m)) \;=\; \mathrm{E}_{h\sim\mathcal{H}}[\delta(h(D_y),\mathcal{U}_m)]. \tag{3}$$

It follows from Equation 3 that

$$\begin{aligned}
\mathrm{E}_{y\sim\{0,1\}^n}[\delta((\mathcal{H},\mathcal{H}(D_y)),(\mathcal{H},\mathcal{U}_m))] &= \mathrm{E}_{y\sim\{0,1\}^n}[\mathrm{E}_{h\sim\mathcal{H}}[\delta(h(D_y),\mathcal{U}_m)]] \\
&= \mathrm{E}_h[\mathrm{E}_y[\delta(h(D_y),\mathcal{U}_m)]] \\
\text{(Using Equation 2)} &\geq \mathrm{E}_h[(1-2\sqrt{\gamma}-2^{-0.1m})(1-2\sqrt{\gamma})] \\
&= (1-2\sqrt{\gamma}-2^{-0.1m})(1-2\sqrt{\gamma}).
\end{aligned}$$

In particular, there exists $y \in \{0,1\}^n$ such that

$$\delta((\mathcal{H},\mathcal{H}(D_y)),(\mathcal{H},\mathcal{U}_m)) \geq (1-2\sqrt{\gamma}-2^{-0.1m})(1-2\sqrt{\gamma}),$$

which completes the proof of Lemma 4.

We proceed now to the proof of our initial claim. Fix a function $h \in \mathcal{H}$. By the definition of noise sensitivity and our assumption on $\mathcal{H}$, for every function $h_i\colon \{0,1\}^n \to \{0,1\}$ obtained from a function $h \in \mathcal{H}$ as the projection of the i-th output bit, we have

$$\Pr_y[h_i(D_y) \neq h_i(y)] \;=\; \Pr_y[h_i(y \oplus X) \neq h_i(y)] \;\leq\; \gamma.$$

Using the linearity of expectation, we obtain

$$\mathrm{E}_y[\Delta(h(D_y),h(y))] \;\leq\; \gamma.$$

By Markov's inequality,

$$\Pr_y[\Delta(h(D_y),h(y)) \leq 1/4] \;\geq\; 1-4\gamma.$$

Using an averaging argument, with probability at least $1-\sqrt{4\gamma}$ over the choice of y, we have that

$$\Pr[\Delta(h(D_y),h(y)) \leq 1/4] \;\geq\; 1-\sqrt{4\gamma}. \tag{4}$$

For any fixed y, consider the following statistical test,

$$T_y \overset{\text{def}}{=} \{z \in \{0,1\}^m \mid \Delta(z,h(y)) \leq 1/4\}.$$

The probability that $\mathcal{U}_m \in T_y$ can be upper bounded via a standard inequality by

$$\Pr_{z \sim \mathcal{U}_m}[z \in T_y] \;\leq\; \frac{2^{m \cdot H_2(1/4)}}{2^m} \;\leq\; 2^{-0.1m}, \tag{5}$$

where $H_2\colon [0,1] \to [0,1]$ is the binary entropy function, and we use the fact that $H_2(1/4) \leq 0.9$. Combining Equations 4 and 5, we get

$$\Pr_y\left[(\Pr_X[h(D_y) \in T_y] - \Pr_{z \sim \mathcal{U}_m}[z \in T_y]) \;\geq\; 1 - \sqrt{4\gamma} - 2^{-0.1m}\right] \;\geq\; 1 - \sqrt{4\gamma},$$

which implies that

$$\Pr_y[\delta(h(D_y), \mathcal{U}_m) \geq 1 - 2\sqrt{\gamma} - 2^{-0.1m}] \;\geq\; 1 - 2\sqrt{\gamma}.$$

Finally, since $\delta(\cdot)$ is non-negative and $\gamma \leq 1/4$, it follows that

$$\mathrm{E}_y[\delta(h(D_y), \mathcal{U}_m)] \;\geq\; (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}),$$

which completes the proof of the claim.

We are now ready to prove a lower bound on the negation complexity of strong extractors.

Proposition 14. *Let $0 < \alpha < 1/2$ be a constant, and $m(n) \geq 100$. Further, suppose that $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ is a family of functions such that each output bit $h_i\colon \{0,1\}^n \to \{0,1\}$ of a function $h \in \mathcal{H}$ is computed by a circuit with t negations. Then, if $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor,*

$$t \;\geq\; \alpha \log n - O(1).$$

Proof. It is known that for any monotone function $g\colon \{0,1\}^n \to \{0,1\}$ and $p(n) \in (0, 1/2)$, $\mathsf{NS}_p(g) = O(\sqrt{n} \cdot p)$ (see e.g. O'Donnell [31]). Using an argument similar to the one employed in the proof of Proposition 13, it follows from Lemma 1 that if $f\colon \{0,1\}^n \to \{0,1\}$ is a Boolean function computed by a circuit with t negations, then

$$\mathsf{NS}_p(f) \;\leq\; C_1 \cdot 2^t \sqrt{n} \cdot p \;\stackrel{\text{def}}{=}\; \gamma,$$

where $C_1 > 0$ is a fixed constant. In other words, this upper bound on the noise sensitivity and our assumption on $\mathcal{H}$ allow us to apply Lemma 4 with an appropriate choice of parameters, which we describe next.

We choose a $0 \leq p \leq \frac{1}{2}$ such that $n \cdot \log \frac{1}{(1-p)} = n^{\frac{1}{2}-\alpha}$. Observe that we can take $p \leq C_2 n^{-\frac{1}{2}-\alpha}$, for an appropriate constant $C_2 > 0$. Let C_3 be a sufficiently large constant such that $C_1 C_2 2^{-C_3} < 1/64$, and suppose that $t < \alpha \log n - C_3$. For this setting of parameters, we obtain

$$\gamma \;=\; C_1 \cdot 2^t \cdot \sqrt{n} \cdot p \;<\; \frac{1}{64}.$$

By Lemma 4, there exists a distribution D of min-entropy $H_\infty(D) = n \log \frac{1}{1-p} = n^{\frac{1}{2}-\alpha}$ for which

$$\begin{aligned} \delta((\mathcal{H}, \mathcal{H}(D)), (\mathcal{H}, \mathcal{U}_m)) &\geq (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}) \\ &> (\frac{3}{4} - 2^{-0.1m}) \cdot \frac{3}{4} \geq \frac{1}{2}, \end{aligned}$$

which contradicts our assumption that $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor. Therefore,

$$t \geq \alpha \log n - C_3 = \alpha \log n - O(1),$$

as desired.

Observe that Proposition 14 provides an almost tight lower bound on the number of negations for extractors with rather weak parameters: in order to extract from reasonable sources only 100 bits that are not ridiculously far from uniform, the corresponding circuits need $\Omega(\log n)$ negations.

5.7 Negations at the Bottom Layer and Circuit Lower Bounds

In this section we solve a problem posed by Koroth and Sarma [25]. Our proof is inspired by ideas introduced in their paper. Our main contribution is the the following general proposition.

Proposition 15. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function, and C be a circuit computing f with negation gates at the bottom layer only. Then,*

$$\mathsf{depth}(C) + \mathsf{negations}(C) \geq \mathsf{depth}^+(f).$$

Proof. Let $d \overset{\text{def}}{=} \mathsf{depth}(C)$, and $t \overset{\text{def}}{=} \mathsf{negations}(C)$. The idea is to use C, a non-monotone circuit for f, to solve the corresponding monotone Karchmer-Wigderson game of f with communication at most $d + t$. It follows from Proposition 1 that $\mathsf{depth}^+(f) \leq d + t$, which completes the proof. More details follow.

Recall that in the monotone Karchmer-Wigderson game for f, Alice is given a string $x \in f^{-1}(1)$, Bob is given $y \in f^{-1}(0)$, and their goal is to agree on a coordinate i such that $x_i = 1$ and $y_i = 0$. Let $T \subset [n]$ be the set of variables that occur negated in C, where $|T| = t$. Given a string $x \in \{0,1\}^n$, we write x_T to denote the substring of x obtained by concatenating the bits indexed by T. During the first round of the protocol, Alice sends x_T to Bob. If among these coordinates there is an index $i \in T$ for which $x_i = 1$ and $y_i = 0$, the protocol terminates with a correct solution. Otherwise, Bob defines a new input $y' \in \{0,1\}^n$ for him as follows: $y'_j \overset{\text{def}}{=} x_j$ if $j \in T$, otherwise $y'_j \overset{\text{def}}{=} y_j$. For convenience, Alice sets $x' \overset{\text{def}}{=} x$.

It is not hard to see that if there was no good index $i \in T$, then $f(x') = 1$ and $f(y') = 0$. Clearly, $1 = f(x) = f(x')$, since $x = x'$. On the other hand, if there is no good index i, y' is obtained from y simply by flipping some bits of

y from 1 to 0. In other words, $y' \preceq y$, and the monotonicity of f implies that $f(y') \leq f(y) = 0$.

Crucially, the players now have inputs $x', y' \in \{0,1\}^n$ that agree on every bit indexed by T. Therefore, without any communication, they are able to simplify the original circuit C in order to obtain a *monotone* circuit $\tilde{C}$ with input variables indexed by $[n]\backslash T$. Let $\tilde{x} \overset{\text{def}}{=} x'_{[n]\backslash T}$ and $\tilde{y} \overset{\text{def}}{=} y'_{[n]\backslash T}$ be the corresponding projections of x' and y'. Clearly, $\tilde{C}(\tilde{x}) = C(x') = f(x') = 1$, and $\tilde{C}(\tilde{y}) = C(y') = f(y') = 0$. Furthermore, $\tilde{C}$ computes some monotone function $\tilde{f} \colon \{0,1\}^{[n]\backslash T} \to \{0,1\}$.

Alice and Bob simulate together the standard Karchmer-Wigderson protocol Π granted by Proposition 1, and obtain an index $j \in [n]\backslash T$ for which $\tilde{x}_j = 1$ and $\tilde{y}_j = 0$. Observe that this stage can be executed with communication cost $\mathsf{depth}(\tilde{C}) \leq \mathsf{depth}(C) = d$. However, since x agrees with $\tilde{x}$ on every bit indexed by $[n]\backslash T$, and similarly for y and $\tilde{y}$, it follows that $x_j = 1$ and $y_j = 0$. Put another way, Alice and Bob have solved the monotone Karchmer-Wigderson game for f with communication at most $t + d$, which completes the proof of our result.

An interesting aspect of this proof is that it relies on both directions of the Karchmer-Wigderson connection. Proposition 15 and previous work on monotone depth lower bounds provide a trade-off between circuit depth and negation complexity for DeMorgan circuits solving the clique problem.

Proposition 16 (Raz and Wigderson [33]). *Let k-*Clique*: $\{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the Boolean function that is 1 if and only if the input graph G contains a clique of size k. If C is a monotone circuit that computes k-*Clique *for $k = n/2$, then* $\mathsf{depth}(C) \geq \gamma \cdot n$*, where $\gamma > 0$ is a fixed constant.*

Corollary 1. *There exists a fixed constant $\gamma > 0$ for which the following holds. If $\delta + \varepsilon \leq \gamma$, then any DeMorgan circuit of depth δn solving the $(n/2)$-*Clique *problem on n-vertex graphs contains at least εn negation gates.*

This result indicates that negation gates at the bottom layer are much easier to handle from the point of view of complexity theory than negations located at arbitrary positions of the circuit (see also Proposition 17 in Section A).

6 Open Problems and Further Research Directions

While our results provide some strong (indeed, optimal) bounds, they also leave open surprisingly basic questions.

For example, it seems reasonable, in light of our results, to think that most cryptographic primitives require $\Omega(\log n)$ negations. Nevertheless, for a basic primitive like a pseudorandom generator (that cannot be monotone), we leave open the following question: Is there a pseudorandom generator computed with a single negation gate? We stress that our question refers to a single circuit with multiple output bits computing the PRG. If one can use different circuits for distinct output bits, then the work of Applebaum, Ishai, and Kushilevitz [4] provides strong evidence that there are PRGs computed with a constant number of negations.

Having negation gates at the bottom level may be easier to study, and with some work we can show (in results omitted from the current paper) that no function with large enough stretch computed with a single negation at the bottom layer can be a small-bias generator (and thus not a pseudorandom generator either).

Another important open problem relates to the negation complexity of WPRF (weak pseudorandom functions, cf. Akavia et al. [1]), or, viewed from the learning perspective, weak-learning functions computed with a single negation. While for strong PRFs, even non-adaptive ones, we have obtained an $\Omega(\log n)$ lower bound, as far as we know, there may exist WPRFs computed by circuits with a single negation gate. Again, when restricting ourselves to negations at the bottom, we can prove some partial results (it is not hard to prove that a function computed by a circuit with a constant number of negations at the bottom layer cannot be a WPRF).

Finally, we have not imposed additional restrictions on the structure of Boolean functions computing cryptographic primitives. For instance, due to efficiency concerns, it is desirable that such circuits have depth as small as possible, without compromising the security of the underlying primitive. It is known that Markov's upper bound of $O(\log n)$ negations fails under restrictions of this form (cf. Santha and Wilson [34]; see also Hofmeister [21]). In particular, this situation sheds some light into why practical implementations have far more negations (or XORs) when compared to the theoretical lower bounds described in our work. Here we have not investigated this phenomenon, and it would be interesting to see if more specific results can be obtained in the cryptographic context.

Acknowledgments. We would like to thank Ilan Orlov for helpful conversations during an early stage of this work, Rocco Servedio for suggesting an initial construction in Proposition 10, and Andrej Bogdanov for helpful discussions that allowed us to extend some results. We also thank the referees for several suggestions that allowed us to improve the presentation.

References

1. Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $\mathrm{AC}^0 \circ \mathrm{MOD}_2$. In *Innovations in Theoretical Computer Science* (ITCS), pages 251–260, 2014.
2. Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley-Interscience, 2008.
3. Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most (1/6)log log n negation gates. *SIAM J. Comput.*, 35(1):201–216, 2005.
4. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

5. Sanjeev Arora and Boaz Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, 2009.
6. Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. More on the complexity of negation-limited circuits. In *Symposium on Theory of Computing* (STOC), pages 585–595, 1995.
7. Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. On the complexity of negation-limited boolean networks. *SIAM J. Comput.*, 27(5):1334–1347, 1998.
8. Eric Blais, Clement C. Canonne, Igor C. Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. *Preprint*, 2014.
9. Avrim Blum, Carl Burch, and John Langford. On learning monotone boolean functions. In *Symposium on Foundations of Computer Science* (FOCS), pages 408–415, 1998.
10. Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *Innovations in Theoretical Computer Science* (ITCS), pages 553–560, 2013.
11. Nader H. Bshouty and Christino Tamon. On the fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996.
12. Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform hardness amplification in NP via monotone codes. *Electronic Colloquium on Computational Complexity* (ECCC), 13(154), 2006.
13. Dana Dachman-Soled, Homin K. Lee, Tal Malkin, Rocco A. Servedio, Andrew Wan, and Hoeteck Wee. Optimal cryptographic hardness of learning monotone functions. *Theory of Computing*, 5(1):257–282, 2009.
14. Michael J. Fischer. The complexity of negation-limited networks - A brief survey. In *Automata Theory and Formal Languages*, pages 71–82, 1975.
15. Cees M. Fortuin, Pieter W. Kasteleyn, and Jean Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971.
16. Mikael Goldmann and Alexander Russell. Spectral bounds on general hard-core predicates. In *Symposium on Theoretical Aspects of Computer Science* (STACS), pages 614–625, 2000.
17. Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2007.
18. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
19. Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory of Computing*, 8(1):231–238, 2012.
20. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Symposium on Theory of Computing* (STOC), pages 25–32, 1989.
21. Thomas Hofmeister. The power of negative thinking in constructing threshold circuits for addition. In *Structure in Complexity Theory Conference*, pages 20–26, 1992.
22. Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012.
23. Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions. In *Symposium on Foundations of Computer Science* (FOCS), pages 68–80, 1988.
24. Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Symposium on Theory of Computing* (STOC), pages 539–550, 1988.

25. Sajin Koroth and Jayalal Sarma. Depth lower bounds against circuits with sparse orientation. In *International Conference on Computing and Combinatorics* (COCOON), pages 596–607, 2014.
26. Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.
27. A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958.
28. Hiroki Morizumi. Limiting negations in formulas. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 701–712, 2009.
29. Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theoretical Computer Science*, 410(38-40):3988–3994, 2009.
30. Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.
31. Ryan O'Donnell. *Analysis of Boolean Functions.* Cambridge University Press, 2014.
32. Ryan O'Donnell and Karl Wimmer. Kkl, kruskal-katona, and monotone nets. *SIAM J. Comput.*, 42(6):2375–2399, 2013.
33. Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.
34. Miklos Santha and Christopher B. Wilson. Limiting negations in constant depth circuits. *SIAM J. Comput.*, 22(2):294–302, 1993.
35. Shao Chin Sung and Keisuke Tanaka. An exponential gap with the removal of one negation gate. *Inf. Process. Lett.*, 82(3):155–157, 2002.
36. Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of Markov's theorem. *Inf. Process. Lett.*, 90(1):15–20, 2004.
37. Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.
38. Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

A The Flow of Negations in Boolean Circuits

In this section we discuss how to move negations in a Boolean circuit in order to explore different aspects of these gates.

A.1 Moving Negations to the Top of the Circuit

For convenience of the reader, we include here a proof for the following structural result about negation gates.

Lemma 5 (Blais et al. [8]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C with at most t negations. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is monotone, $T = O(2^t)$, and h is either the parity function, or its negation.*

Proof. Recall from Proposition 3 that if a Boolean function f is computed by a circuit with t negations, then $k \overset{\text{def}}{=} a(f) \leq O(2^t)$. The claimed result follows easily once we "f-slice" the boolean hypercube, as described next. For any $i \in \{0, 1, \ldots, k\}$, let

$$T_i = \{x \in \{0,1\}^n \mid a(f, Y) \leq i, \text{ for any chain } Y \text{ starting at } x\}.$$

Observe that T_i is a monotone set: if $x \in T_i$ and $x \preceq y$, then $y \in T_i$. In addition, it is clear that $T_0 \subset T_1 \subset \ldots \subset T_k = \{0,1\}^n$. Finally, for every $i \in \{0, 1, \ldots, k-1\}$, it is not hard to see that $T_{i+1} \setminus T_i \neq \emptyset$, since $k = a(f)$.

Let $S_0 \overset{\text{def}}{=} T_0$. Observe that $1^n \in T_0$, hence S_0 is nonempty. For any $j \in \{1, \ldots, k\}$, set $S_j \overset{\text{def}}{=} T_j \setminus T_{j-1}$. It follows from the preceding discussion that the family $\mathcal{S} = \{S_0, S_1, \ldots, S_k\}$ is a partition of the n-dimensional boolean cube into nonempty sets.

Next, we prove the following claim: There exists a vector $b = (b_0, b_1, \ldots, b_k) \in \{0,1\}^{k+1}$ with the following properties:

(*i*) For each $0 \leq i \leq k$, and for any $x^i \in S_i$, we have $f(x^i) = b_i$.
(*ii*) For each $0 < i \leq k$, we have $b_i = 1 - b_{i-1}$.
(*iii*) For each $0 \leq i \leq k$, given any $x^i \in S_i$, there exist elements $x^0, x^1, \ldots, x^{i-1}$ in $S_0, S_1, \ldots, S_{i-1}$, respectively, such that $Y = (x^i, x^{i-1}, \ldots, x^0)$ is a chain starting at x^i with $a(f, Y) = i$. Further, every proper chain Y starting at x^i with $a(f, Y) = i$ is of this form, i.e., its elements belong to distinct sets S_j, where $j < i$.

The construction of this vector and the proof of these items is by induction on $i \in [0, k]$. For $i = 0$, we set $b_0 = f(1^n)$, and observe that the result is true using the definition of S_0 and T_0. Consider now an element $y^i \in S_i$, where $i > 0$, and assume that items (*i*), (*ii*), and (*iii*) hold for any smaller index. Since $y^i \in S_i$, there exists a proper chain $Y = (y^i, y^{i-1}, \ldots, y^0)$ with $a(f, Y) = i$. Since this chain cannot be extended to a larger chain starting at y^i, it follows from the induction hypothesis that, for every $0 < j \leq i$, $f(y^j) = 1 - f(y^{j-1}) = 1 - b_{j-1}$. In particular, we can set $b_i = 1 - b_{i-1}$, since our initial element $y^i \in S_i$ was arbitrary. Finally, the remaining part of item (*iii*) follows once we consider the subchain $Y' = (y^{i-1}, \ldots, y^0)$, and apply the induction hypothesis.

Finally, we use items (*i*) and (*ii*) to prove the lemma. Recall that every family T_i is monotone, where $i \in \{0, 1, \ldots, k\}$. In other words, there exist $k + 1$ monotone functions $g_i \colon \{0,1\}^n \to \{0,1\}$ such that $g_i^{-1}(1) = T_i$, where $i \in [0, k]$. As observed before, $S_k = T_k \setminus T_{k-1}$ is nonempty. In particular, $0^n \in S_k$. If $f(0^n) = 0$, we let $h \colon \{0,1\}^{k+1} \to \{0,1\}$ be the parity function $\sum_{j=0}^{k} y_i \pmod 2$. Otherwise, we let h be the complement of the parity function. It follows from (*i*) and (*ii*) that, for every $x \in \{0,1\}^n$, we have $f(x) = h(g_0(x), \ldots, g_k(x))$, which completes the proof.

By relaxing the assumption on h, we can prove the following effective version of Lemma 1.

Lemma 6. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s containing t negation gates, where $t \geq 0$. Then f can be written as*

$f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is computed by a monotone circuit of size at most s, $T = 2^{t+1} - 1$, and $h\colon \{0,1\}^T \to \{0,1\}$ is computed by a circuit of size at most $5T$.

Proof. The proof is by induction on t. The base case $t = 0$ is trivial. Now let $t \geq 1$, and assume the statement holds for any function computed by circuits with at most $t' < t$ negations. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s that contains t negations. Let $x_1, \ldots x_n, f_1, \ldots, f_s$ be the functions computed at each internal node of C, and $\lceil f_1 \rceil, \ldots, \lceil f_s \rceil$ be the corresponding gates, i.e., each $\lceil f_i \rceil \in \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$. Furthermore, assume that this sequence is a topological sort of the nodes of the circuit, in the sense that the inputs of each gate $\lceil f_i \rceil$ are $f_{i_1}(x)$ and $f_{i_2}(x)$, with $i_1, i_2 < i$. Let $i^* \in [s]$ be the index of the first NOT gate in C according to this sequence.

Consider a new circuit C' over $n+1$ variables $x_1, \ldots, x_n, y$, where C' computes exactly as C, except that the output value of f_{i^*} is replaced by the new input y. By construction, C' is a circuit of size at most s containing $t' \overset{\text{def}}{=} t-1$ negations, and it computes some Boolean function $f'\colon \{0,1\}^{n+1} \to \{0,1\}$. Applying the induction hypothesis, we get that

$$f'(\boldsymbol{x}, y) = h'(g'_1(\boldsymbol{x}, y), \ldots, g'_{T'}(\boldsymbol{x}, y)), \tag{6}$$

where each g'_i is computed by a monotone circuit of size at most s, $T' \leq 2^{t'+1} - 1$, and $h'\colon \{0,1\}^{T'} \to \{0,1\}$ admits a circuit of size $5T'$. In addition, notice that

$$f(\boldsymbol{x}) = \begin{cases} f'(\boldsymbol{x}, 1) & \text{if} \quad f_{i^*}(\boldsymbol{x}) = 1, \\ f'(\boldsymbol{x}, 0) & \text{otherwise.} \end{cases} \tag{7}$$

Let f_i be the input wire of $\lceil f_{i^*} \rceil$. Since $\lceil f_{i^*} \rceil = \mathsf{NOT}$, we obtain using Equation 7 that

$$f(\boldsymbol{x}) = \tilde{h}(f_i(\boldsymbol{x}), f'(\boldsymbol{x}, 0), f'(\boldsymbol{x}, 1)), \tag{8}$$

where $\tilde{h}(z, y_1, y_0) \overset{\text{def}}{=} y_z$ is a function over three input bits that is computed by a circuit of size at most 5. Furthermore, combining Equations 6 and 8, it follows that

$$\begin{aligned} f(\boldsymbol{x}) &= \tilde{h}(f_i(\boldsymbol{x}), h'(g'_1(\boldsymbol{x}, 0), \ldots, g'_{T'}(\boldsymbol{x}, 0)), h'(g'_1(\boldsymbol{x}, 1), \ldots, g'_{T'}(\boldsymbol{x}, 1))) \\ &= h(f_i(\boldsymbol{x}), g'_{1,0}(\boldsymbol{x}), \ldots, g'_{T',0}(\boldsymbol{x}), g'_{1,1}(\boldsymbol{x}), \ldots, g'_{T',1}(\boldsymbol{x})), \end{aligned}$$

where $g'_{j,b}(\boldsymbol{x}) \overset{\text{def}}{=} g'_j(\boldsymbol{x}, b)$, for $j \in [T']$ and $b \in \{0,1\}$, and $h\colon \{0,1\}^{2T'+1} \to \{0,1\}$ is the function obtained by setting

$$h(v_0, v_1, \ldots, v_{T'}, v_{T'+1}, \ldots, v_{2T'}) \overset{\text{def}}{=} \tilde{h}(v_0, h'(v_1, \ldots, v_{T'}), h(v_{T'+1}, \ldots, v_{2T'})).$$

Using our assumption on i^*, it follows that f_i is computed by a monotone circuit of size s. It is also clear that each $g'_{j,b}$ admits a monotone circuit of size s. Further, observe that

$$2T' + 1 \leq 2(2^{t'+1} - 1) + 1 = 2(2^t - 1) + 1 = 2^{t+1} - 1 \overset{\text{def}}{=} T.$$

Finally, using the induction hypothesis and the upper bound on the circuit size of $\tilde{h}$, we get that h is computed by a circuit of size at most

$$5 + 5T' + 5T' = 5(2T' + 1) = 5T,$$

which completes the proof of Lemma 2.

It is possible to show that the upper bound on T in the statement of Lemma 7 is essentially optimal. This follows from the connection between the number of negation gates in a Boolean circuit for a function f and the alternation complexity of f, as discovered by Markov [27] (see e.g. Blais et al. [8] for further details).

A.2 Moving Negations to the Bottom of the Circuit

We recall the following basic fact about negations.

Fact 1 *Let C be a Boolean circuit of size s containing a negation gate at depth $d \geq 1$. Then C can be transformed into an equivalent circuit C' of size s without this negation gate that contains at most 2^{d-1} additional negations at the bottom layer.*

Proof. The result is immediate from the application of DeMorgan rules for Boolean connectives.

We observe below that this result is optimal. Put another way, a negation gate at an arbitrary location can be more powerful than a linear number of negations at the bottom layer.

Proposition 17. *There exists an explicit Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that admits a linear size circuit C containing a single negation gate, but for which any equivalent circuit C' with negation gates at the bottom layer only requires n negations.*

Proof. Let $f(x) = 1$ if and only if $x = 0^n$. Clearly, f can be computed by a circuit with a single negation, since this function is the negation of a monotone function. The lower bound follows using an argument from [25]. Assume that $f(x) = D(x, (x \oplus \beta))$, where D is a monotone circuit. We need to prove that $\beta_i = 1$ for every $i \in [n]$. Consider inputs $z \overset{\text{def}}{=} 0^n$ and $e_i \overset{\text{def}}{=} 0^{i-1}10^{n-i}$. By definition, $f(z) = 1$ and $f(e_i) = 0$, thus $D(0^n, 0^n \oplus \beta) = D(0^n, \beta) = 1$ and $D(e_i, e_i \oplus \beta) = D(e_i, \beta^{\oplus i}) = 0$. If $\beta_i = 0$, then $(0^n, \beta) \prec (e_i, \beta^{\oplus i})$, and since D is monotone, we get $D(0^n, \beta) \leq D(e_i, \beta^{\oplus i})$. However, this is in contradiction with the value of f on z and e_i, which implies that $\beta_i = 1$.