Black-Box Obfuscation for d-CNFs

Zvika Brakerski Guy N. Rothblum

Abstract

We show how to securely obfuscate a new class of functions: *conjunctions of $\mathcal{NC}^0_d$ circuits*. These are functions of the form $C(\vec{x}) = \bigwedge_{i=1}^{m} C_i(\vec{x})$, where each C_i is a boolean $\mathcal{NC}^0_d$ circuit, whose output bit is only a function of $d = O(1)$ bits of the input $\vec{x}$. For example, d-CNFs, where each clause is a disjunction of at most d variables, are in this class. Given such a function, we produce an obfuscated program that preserves the input-output functionality of the given function, but reveals nothing else. Our construction is based on multilinear maps, and can be instantiated using the recent candidates proposed by Garg, Gentry and Halevi (EUROCRYPT 2013) and by Coron, Lepoint and Tibouchi (CRYPTO 2013).

We prove that the construction is a secure obfuscation in a generic multilinear group model, under the black-box definition of Barak et al. (CRYPTO 2001). Security is based on a new *worst-case* hardness assumption about exponential hardness of the NP-complete problem 3-SAT, the *Bounded Speedup Hypothesis*.

One of the new techniques we introduce is a method for enforcing input consistency, which we call *randomizing sub-assignments*. We hope that this technique can find further application in constructing secure obfuscators.

The family of functions we obfuscate is considerably richer than previous works that consider black-box obfuscation.[1] As one application, we show how to achieve *obfuscated functional point testing*: namely, to construct a circuit that checks whether $f(\vec{x}) = \vec{y}$, where f is an arbitrary "public" polynomial-time computable function, but $\vec{y}$ is a "secret" point that is hidden in the obfuscation.

1 Introduction

A code obfuscator is a compiler that makes computer programs impossible to reverse engineer, while preserving their input-output functionality. Code obfuscation is a central question in the theory and practice of cryptography. Despite the problem's importance, very few techniques or heuristics were known. Recently, however, several works have leveraged new constructions of cryptographically secure multilinear maps [GGH12, CLT13] to construct obfuscators for complex functionalities [BR13, GGH$^+$13].

In this work, we present a candidate obfuscator for a rich new circuit class: conjunctions of $\mathcal{NC}^0$ circuits. In particular, this includes the family of d-CNF formulas (for constant d). Our construction relies on *(asymmetric) multilinear maps*, and can be instantiated using the new candidate constructions of Garg, Gentry and Halevi [GGH12], or of Coron, Lepoint and Tibouchi [CLT13].

We prove the construction's security in a generic multilinear map model, under a worst-case complexity assumption (see below). An important ingredient in this construction, which we hope

[1] A concurrent and independent work of Garg et al. [GGH$^+$13] considers the weaker security notion of "indistinguishability obfuscation", and proposes a candidate obfuscator for any function computed by a polynomial size circuit.

will find further applications, is a new technique for enforcing input consistency using *randomizing sub-assignments*.

Obfuscation: Definitions. Intuitively, an obfuscator should generate a new program that preserves the the original program's functionality, but is impossible to reverse engineer. The theoretical study of this problem was initiated by Barak et al. [BGI+12]. They formalized a strong simulation-based security requirement of *black box obfuscation*: namely, the obfuscated program should expose nothing more than what can be learned via oracle access to its input-output behavior. We refer to this notion as "black-box" obfuscation, and we use this strong formalization throughout this work (except where we note otherwise).

A weaker notion of obfuscation, known as *indistinguishability* or *best-possible* obfuscation was studied in [BGI+12, GR07]. An indistinguishability obfuscator guarantees that the obfuscations of any two programs (boolean circuits) with identical functionalities are indistinguishable. We note that, unlike the black-box definition of security, indistinguishability obfuscation does not quantify or qualify what information the obfuscation might expose. In particular, the obfuscation might reveal non-black-box information about the functionality.

Prior Work: Negative Results. In their work, [BGI+12] proved the impossibility of general-purpose black-box obfuscators (i.e. ones that work for any polynomial-time functionality) in the virtual black box model. This impossibility result was extended in [GK05]. While these negative results show serious limitations on the possibility of general-purpose obfuscation, they focus on specific (often cryptographic or contrived) functionalities. Thus, they do not rule out that obfuscation may be possible for many programs of interest. Goldwasser and Rothblum [GR07] showed obstacles to the possibility of achieving indistinguishability obfuscation with information-theoretic security, and to achieving it in the random oracle model.

Prior Work: Positive Results. Positive results on obfuscation focused on specific, simple programs. One program family, which has received extensive attention, is that of "point functions": password checking programs that only accept a single input string, and reject all others. Starting with the work of Canetti [Can97], several works have shown obfuscators for this family under various assumptions [CMR98, LPS04, Wee05], as well as extensions [CD08, BC10]. Canetti, Rothblum and Varia [CRV10] showed how to obfuscate a function that checks membership in a hyperplane of constant dimension (over a large finite field). Other works showed how to obfuscate cryptographic function classes under different definitions and formalizations. These function classes include checking proximity to a hidden point [DS05], vote mixing [AW07], and re-encryption [HRSV11]. Several works [Can97, CMR98, HMLS10, HRSV11] relaxed the security requirement so that obfuscation only holds for a random choice of a program from the family.

More recently, Brakerski and Rothblum [BR13] showed that multilinear maps could be used to obfuscate richer function families. They constructed an obfuscator for *conjunctions*, the family of functions that test whether a subset of the input bits take on specified values. The construction was shown to be secure in the generic multilinear map model. It was also shown to be secure under (falsifiable [Nao03]) multilinear DDH-like assumptions, so long as the conjunction is drawn from a family with sufficient entropy.

Related Concurrent Work [GGH$^+$13]. In a concurrent and independent work, Garg et al. [GGH$^+$13] use cryptographic multilinear maps to give a candidate indistinguishability obfuscator (see above) for all polynomial-size circuits. The main differences with our work are: (*i*) we construct an obfuscator with the stronger security notion of black-box obfuscation (albeit for a smaller class of functions). And (*ii*) we prove the security of our construction in the generic multilinear group model. See more on the generic multilinear group model below. Our proof uses a worst-case complexity-theoretic assumption. Garg et al. [GGH$^+$13] pose the security of their construction in the generic multilinear group model as a major open question. We note that [GGH$^+$13] do prove security in a new "generic colored matrix model", which restricts an adversary to a subset of the multilinear group operations.

Our Work: Black-Box Obfuscation for Conjunctions of $\mathcal{NC}^0$. An $\mathcal{NC}^0_d$ circuit is a boolean circuit on n-bit inputs, whose output depends only on d input bits. If a circuit (or rather a circuit ensemble) is in $\mathcal{NC}^0_d$ for some fixed $d \geq 0$, then we say that it is in $\mathcal{NC}^0$ (and we drop the d). A *conjunction of $\mathcal{NC}^0$ circuits* is a circuit $C = \bigwedge_{i=1}^{m} C_i$, where each C_i is an $\mathcal{NC}^0$ circuit. We use $\mathcal{CNC}^0$ to refer to the class of circuits that are conjunctions of $\mathcal{NC}^0$ circuits. For example, any d-CNF (a CNF formula, where each clause has at most d literals) is a $\mathcal{CNC}^0$ circuit. Another example is simple conjunction circuits: there each C_i only depends on (at most) a single input bit.

In this work we build a black-box obfuscator for $\mathcal{CNC}^0$ circuits. In more detail, the obfuscator takes as input a $\mathcal{CNC}^0_d$ circuit C, runs in time $O(n^d)$, and outputs and obfuscation that exposes nothing beyond the input-output functionality of C (the obfuscation also exposes d). We note that the class $\mathcal{CNC}^0$ is limited in computational power. In particular, it is contained in $\mathcal{AC}^0$, and so there are simple and natural functions (such as XOR) that cannot be computed in $\mathcal{CNC}^0$. Still, it is significantly richer than previous classes for which obfuscators were known. For example, the Cook-Levin Theorem shows that NP-complete statements can be verified in $\mathcal{CNC}^0$, and there are secure digital signature schemes where verification is in $\mathcal{CNC}^0$. Moreover, impossibility results [BGI$^+$12, GK05] rule out general-purpose black-box obfuscation even for $\mathcal{AC}^0$. Obfuscating $\mathcal{CNC}^0$ brings us close to the dovetailing with these impossibility results.

One important ingredient in this construction is a new technique for enforcing input consistency by *randomizing sub-assignments*. We are hopeful that this technique may find further applications.

We prove that the construction is a secure black-box obfuscator in the generic multilinear group model. In the generic multilinear group model, an adversary must operate independently of group elements' representations. Rather, the adversary is given arbitrary strings representing these elements, and can only manipulate them using oracles for addition, subtraction, multilinear maps and more (see below). In our proof, we make use of a new *worst-case* assumption about exponential hardness of the NP-complete problem 3-SAT. We call this new assumption the *bounded speedup hypothesis*, a strengthening of the long-standing exponential time hypothesis (ETH) for solving SAT [IP99].[2] See further details below.

We find this proof of security intriguing for several reasons. First, as stated above, it tells us that (under the bounded-speedup hypothesis) the construction is completely impervious to generic attacks—a rich class that of attacks, which in particular includes all known attacks against the candidate cryptographic multilinear groups. While we distrust the generic model (especially in light of impossibility results [BGI$^+$12, GK05]), results in this model can still be used to obfuscate

[2]We note that is both the adversary and the simulator are allowed to run in quasi-polynomial tim, security can be based on the (standard) Exponential-Time Hypothesis.

provided we have simple secure hardware implementing the generic group. Finally, to the best of our knowledge, no impossibility results for black-box obfuscation are known in generic models (or in the random oracle model). This raises the possibility that our techniques could be used to construct a secure *black-box* obfuscator for richer classes, and even for any polynomial-size circuit, in the generic multilinear group model.

An Application: Obfuscated Functional Point Tests. Consider the following setting: there is an arbitrary polynomial-size "public" circuit computing a function f, and a "secret" target value $\vec{y}$. We want to release an obfuscation that can test, for any input $\vec{x}$, whether $f(\vec{x}) = \vec{y}$, but *without revealing anything about the secret* $\vec{y}$. For example, perhaps C checks that its input contains a message m, and a digital signature on m from a trusted authority (otherwise C rejects). If this test passes, C runs a sophisticated analysis to determine m's topic, and flags whether m's topic equals y. An adversary that receives the obfuscation (and knows the topic analysis algorithm) should not be able to determine what the "target topic" y is—it can only test messages that have been signed by the trusted authority, and see whether their topic is y.

A obfuscator for $\mathcal{NC}^0$ can be used for this application as follows. By the Cook-Levin theorem, for any polynomial-size C, there exists a $\mathcal{NC}^0_3$ circuit C' that takes as input $(\vec{x}, C(\vec{x}), w)$, where w is "advice" computed as a function of C and of $\vec{x}$ only, and accept if and only if $C(\vec{x}) = y$. By obfuscating C' gives the desired application. We note that this type of application, involving general computations, is well beyond what was known to be possible for black-box obfuscation.

We proceed with further details and discussions.

1.1 Our Construction and its Security

Background on Multilinear Maps and Graded Encoding Schemes. We begin by recalling the notion of multilinear maps, due to Boneh and Silverberg [BS02]. Rothblum [Rot13] considered the asymmetric case, where the groups may be different (this is crucial for our construction).

Definition 1.1 (Asymmetric Multilinear Map [BS02, Rot13])**.** For $\tau + 1$ cyclic groups $G_1, \ldots, G_\tau$, G_T of the same order p, a τ-multilinear map $e : G_1 \times \ldots \times G_\tau \to G$ has the following properties:

1. For elements $\{g_i \in G_i\}_{i=1,\ldots,\tau}$, index $i \in [\tau]$ and integer $\alpha \in \mathbb{Z}_p$, it holds that:
$$e(g_1, \ldots, \alpha \cdot g_i, \ldots, g_\tau) = \alpha \cdot e(g_1, \ldots, g_\tau)$$
2. The map e is non-degenerate: when its inputs are all generators of their respective groups $\{G_i\}$, then its output is a generator of the group G.

Recently, [GGH12] suggested a candidate for graded encoding, a generalization of (symmetric or asymmetric) multilinear maps. See Section 2.1 for a more complete overview on these. For this introduction, we treat them as a generalization of asymmetric multilinear maps in the following way. For the i-th group G_i, of prime order p, we consider the ring $\mathbb{Z}_p$. For an element $\sigma \in \mathbb{Z}_p$, we can think of g_i^σ as an "encoding" of σ in G_i. We denote this by $\mathsf{enc}_i(\sigma)$. We note that this encoding is easy to compute, but (presumably) hard to invert. The multilinear map e lets us take τ encodings $\{\mathsf{enc}_i(\sigma_i)\}_{i\in[\tau],\sigma_i\in\mathbb{Z}_p}$, and compute an encoding $\mathsf{enc}_T(\prod_i \sigma_i)$ in the "final group" G. Graded encoding schemes provide a similar functionality, albeit with randomized and noisy encodings, and with a procedure for testing equality of encoded elements in the final group G.

Obfuscating Conjunctions of Bit-Predicates [BR13]. Recently, [BR13] show how to obfuscate conjunction functions: conjunctions of single-bit predicates. A conjunction $C(\vec{x}) = \bigwedge_{i\in[n]} C_i(\vec{x}[i])$ is made up of a single circuit C_i for each input bit $\vec{x}[i]$. The circuit C_i can equal $\vec{x}[i]$, its negation, or be a constant 1 (meaning it has no effect) or 0 (meaning that $C \equiv 0$). For each $i \in [n]$, the [BR13] obfuscator chooses an "accepting" ring element $\alpha_{i,0}$ and a "rejecting" ring element $\alpha_{i,1}$. It then encodes the circuit C_i using a two-entry table, where each entry $v \in \{0,1\}$ contains *the encodings* of a pair of ring elements:

$$(\rho_{i,v}, (\rho_{i,v}) \cdot \alpha_{i,C_i(v)}))$$

(the $\rho_{i,v}$ value is uniformly random). We emphasize that the obfuscation only produces the *encodings* of these ring elements. Note that e.g. if $C_i \equiv 1$, then both of these table entries use the same α variable, and are thus not independent. Nonetheless, since the encodings are hard to invert, an adversary will not be able to tell that this is the case (which is essential for security).

To complete the obfuscation, the obfuscator also produces encodings of an "unlocking" or "checking" pair:

$$(\rho_{\mathsf{chk}}, (\rho_{\mathsf{chk}} \cdot (\prod_{i\in[n]} \alpha_{i,1})))$$

(where again ρ is uniformly random, and chk is just a notation for the last group, $n+1$ in this case).

To evaluate the obfuscated program on an input $\vec{x} \in \{0,1\}^n$, they use the multilinear map to test the equality:[3]

$$\left(\rho_{\mathsf{chk}} \cdot (\prod_{i\in[n]} (\rho_{i,\vec{x}[i]}) \ \cdot \alpha_{i,C_i(\vec{x}[i])})\right) \stackrel{?}{=} \left((\rho_{\mathsf{chk}} \cdot (\prod_{i\in[n]} \alpha_{i,1})) \cdot (\prod_{i\in[n]} \rho_{i,\vec{x}[i]})\right) \tag{1}$$

Correctness follows by construction: the equality holds if and only if $\forall i \in [n] : C_i(\vec{x}[i]) = 1$ (except for a negligible error probability).

Security is not as straightforward. Intuitively, security holds because for each $i \in [n]$, the entries of the i-th table do not reveal anything about C_i (the complication is that the "checking" pair creates correlations between the different table entries. See [BR13]).

Our Contribution: Obfuscating Conjunctions of Multi-Bit Predicates. Our goal is obfuscating a $\mathcal{CNC}^0$ circuit C, where C is a conjunction of $\mathcal{NC}^0$ predicates $C_1, \ldots, C_m$. A natural initial approach is starting with the construction of [BR13], which obfuscates conjunctions of single-bit predicates, and trying to extend that construction to conjunctions of predicates on many (d) bits as follows. We can "group" every d-tuple of inputs bits into a "super-symbol". This stretches the n-bit input into a vector of $\binom{n}{d}$ "super-symbols" in $\{0,1\}^d$. Now, any predicate on d bits becomes a predicate on a single "super-symbol", and we might hope to directly apply the conjunction obfuscator.

There are two issues with this approach:

1. **Each input "super-symbol" is now in $\{0,1\}^d$ (rather than $\{0,1\}$)**

[3] For the candidate of [GGH12], the encodings are randomized, but there is a procedure for testing equality between encoded elements in the final group.

Examine one such super-symbol I, corresponding to a d-tuple of input bits (i.e. I is a set of d elements out of $[n]$, which we denote $I \in \binom{[n]}{d}$). Let C_I be the circuit in C that takes the d bits in I as its input bits.[4] Of the 2^d assignments to the bits in I, there is a subset $S_1 \subseteq \{0,1\}^d$ that makes C_I accept. The remaining assignments $S_0 \subseteq \{0,1\}^d$, make C_I reject.

Building on the conjunction obfuscator, we can now have a table of size 2^d for the circuit C_I, where each table entry corresponds to an assignment for the variables in I. We choose an "accepting ring element" $\alpha_{I,1}$ and a "rejecting group element" $\alpha_{I,0}$. In each entry $\vec{s} \in \{0,1\}^d$ we encode either an "accepting pair" $(\rho_{I,\vec{s}}, (\rho_{I,\vec{s}} \cdot \alpha_{I,1}))$, or a "rejecting pair" $(\rho_{I,\vec{s}}, (\rho_{I,\vec{s}} \cdot \alpha_{I,0}))$. Intuitively, hardness of discreet log problems guarantees that an adversary cannot use these table to distinguish whether or not any given assignment satisfies the I-th sub-circuit.

Finally, to complete the obfuscation, we also include a "checking pair" that encodes a product of all the accepting group elements $(\rho_{\mathsf{chk}}, (\rho_{\mathsf{chk}} \cdot \prod_{I \in \binom{[n]}{d}} \alpha_{I,1}))$.

To evaluate the obfuscated program on an input $\vec{x} \in \{0,1\}^n$, use the multilinear map, as in the conjunction obfuscator, to test the equality:

$$\left(\rho_{\mathsf{chk}} \cdot \left(\prod_{I \in \binom{[n]}{d}} (\rho_{I,\vec{x}_{|I}} \cdot \alpha_{I,\vec{x}_{|I}}) \right) \right) \stackrel{?}{=} \left(\rho_{\mathsf{chk}} \cdot \left(\prod_{I \in \binom{[n]}{d}} \alpha_{I,1} \right) \cdot \left(\prod_{I \in \binom{[n]}{d}} \rho_{I,\vec{x}_{|I}} \right) \right)$$

By construction, equality will hold if and only if $\vec{x}$ is an accepting input (except for a negligible error probability). The size of the resulting obfuscation is $O(n^d \cdot 2^d)$.

2. **Enforcing Consistency**

   A more serious issue with the idea above, is *enforcing consistency*. Namely, there is nothing to stop an adversary from picking $\binom{n}{d}$ arbitrary assignments $\{\vec{s}_I \in \{0,1\}^d\}_{I \in \binom{[n]}{d}}$, and evaluating the "checking equality":

$$\left(\rho_{\mathsf{chk}} \cdot \left(\prod_{I \in \binom{[n]}{d}} (\rho_{I,\vec{s}_I} \cdot \alpha_{I,\vec{s}_I}) \right) \right) \stackrel{?}{=} \left(\rho_{\mathsf{chk}} \cdot \left(\prod_{I \in \binom{[n]}{d}} \alpha_{I,1} \right) \cdot \left(\prod_{I \in \binom{[n]}{d}} \rho_{I,\vec{s}_I} \right) \right)$$

   We emphasize that *the assignments $\vec{s}_I$ might not be consistent*: they need not correspond to any input $\vec{x} \in \{0,1\}^n$. For example, the first bit of $\vec{x}$ might be set to 0 in one set I, and to 1 in another set I'! Allowing an adversary to test such equalities would immediately leak non black-box information about the circuit C and would break security.

   The main technical challenge in our construction is disallowing such attacks, and enforcing consistency in the adversary's choices. We view this as one of our main contributions, and we hope that it can find further applications as a building-block for constructing secure obfuscators.

[4] We assume w.l.o.g. that there is a single such circuit C_I in C: if there is none, then C_I is the constant-1 circuits. If there are multiple such circuits, they can be merged into their conjunction, which is itself an $\mathcal{NC}^0_d$ circuit. Both transformations maintain the functionality of C and also maintain $|C| \leq O(n)^d$.

Randomizing Sub-Assignments and Enforcing Consistency. We want to avoid letting an adversary choose inconsistent sub-assignments for the circuits $\{C_I\}_{I\in\binom{[n]}{d}}$. In particular, if $i \in I, I'$, then we want to ensure that the adversary can only pick sub-assignments $\vec{s}_I, \vec{s}_{I'}$ where $\vec{s}_I[i] = \vec{s}_{I'}[i] \in \{0,1\}$. Towards this end, we randomize each sub-assignment's table entry as follows.

For each $i \in [n]$ and $I \ni i$ (we use $I \ni i$ to denote $\{I \in \binom{[n]}{d} : i \in I\}$), and for each bit value $b \in \{0,1\}$, choose a uniformly random randomizer $\beta_{I,i,b}$, and multiply it into each entry $\vec{s} \in \{0,1\}^d$ of the table for I, so that entry now has encodings of:

$$(\rho_{I,\vec{s}}, (\rho_{I,\vec{s}} \cdot \alpha_{I,C_I(\vec{s})} \cdot (\prod_{i\in I} \beta_{I,i,\vec{s}[i]})))$$

Now in evaluating the "checking equality", on the left-hand side we have a product of all these β-randomizers, i.e. we get an additional multiplicative term:

$$\prod_{I\in\binom{[n]}{d}, i\in I} \beta_{I,i,\vec{s}_I[i]} = \prod_{i\in[n]} (\prod_{I\ni i} \beta_{I,i,\vec{s}_I[i]})$$

We pick the β-randomizers under the additional restriction that $\forall i \in [n]$:

$$(\prod_{I\ni i} \beta_{I,i,\mathbf{0}}) = (\prod_{I\ni i} \beta_{I,i,\mathbf{1}}) = \gamma_i$$

Otherwise the β variables are uniformly random. This has the effect that as long as the adversary picks the sub-assignments involving variable i *consistently*, the new multiplicative term is exactly γ_i, regardless of whether the bit value assigned to the i-th input bit is 0 or 1. However, if assignments involving input bit i get *inconsistent* values, then the added multiplicative term will be different from γ_i (indeed, it will be close to uniformly random and independent of γ_i).

We use the above randomization to enforce consistency. To allow functionality, we multiply the new terms into the "checking pair", which now has encodings of:

$$(\rho_{\mathsf{chk}}, (\rho_{\mathsf{chk}} \cdot \prod_{I\in\binom{[n]}{d}} \alpha_{I,1} \cdot \prod_{i\in[n]} \gamma_i))$$

The obfuscated program can now be evaluated using the target equation similarly to the above (with the added β terms enforcing consistency). Functionality follows by construction. Security is not so straightforward: while the randomized sub-assignments do intuitively seem to aid in avoiding inconsistent assignments to the sub-circuits, proving that the construction is secure seems to be considerably more difficult.

Security in the Generic Model and the Bounded-Speedup Hypothesis We prove that our construction is a secure black-box obfuscator in the generic multilinear group model. In this proof, we make use of a new *worst-case* assumption about exponential hardness of the NP-complete problem 3-SAT. We call this new assumption the *bounded speedup hypothesis*, a strengthening of the long-standing exponential time hypothesis for solving SAT [IP99]. The exponential-time hypothesis states that no sub-exponential time algorithm can resolve satisfiability of 3-CNF formulas. Intuitively, the bounded-speedup hypothesis states that no polynomial-time algorithm for resolving satisfiability of 3-CNFs can have "super-polynomial speedup" over a brute-force algorithm that

tests assignments one-by-one. More formally, there does not exist an ensemble of polynomial-size circuits $\{\mathcal{A}_n\}$, and an ensemble of super-polynomial-size sets of assignments $\{\mathcal{X}_n\}$, such that on input a 3-CNF Φ on n-bit inputs, w.h.p. $\mathcal{A}_n$ finds a satisfying assignment for Φ in $\mathcal{X}_n$ if such an assignment exists. We emphasize that this is a worst-case hypothesis, i.e. it only says that for every ensemble $\mathcal{A}$, there *exists* some 3-CNF on which $\mathcal{A}$ fails. See Definition 5.3, and further discussion in Appendix A below.

At an intuitive (and somewhat inaccurate) level, the security proof uses the randomizing sub-assignments to argue that the only "interesting" computations that an adversary can run in the generic model are: (i) evaluate the "checking equation" on a consistently chosen input $\vec{x}$, or (ii) evaluate a linear combination of the checking equations for a set $\mathcal{X}$ of inputs. For any other computations that the adversary runs, the answer is essentially independent of the obfuscated circuit C, and can be simulated.

For computations that evaluate a linear combination of the checking equation for inputs in a set $\mathcal{X}$, the answer can be simulated using only black-box access to C on those inputs. The remaining challenge is to argue that the adversary can only evaluate a linear combination of *polynomially* many inputs, i.e. that the set $\mathcal{X}$ must be of polynomial size. To show this, we prove that an adversary that computes a linear combination of the checking equation for a set of inputs $\mathcal{X}$, can also resolve 3-SAT on this same set $\mathcal{X}$: given any 3-CNF Φ, it can find a satisfying assignment for Φ in $\mathcal{X}$ (if one exists). Thus, under the Bounded Speedup Hypothesis, the set $\mathcal{X}$ must be of polynomial size, and the adversary can be simulated. We note that one detail (among many) omitted above, is how the simulator can "extract" the set $\mathcal{X}$ to complete the simulation. See details in Section 5.

2 Preliminaries

For all $n, d \in \mathbb{N}$ we define $\binom{[n]}{d}$ to be the set of lexicographically ordered sets of cardinality d in $[n]$. More formally:

$$\binom{[n]}{d} = \left\{ \langle i_1, \ldots, i_d \rangle \in [n]^d : i_1 < \cdots < i_d \right\} .$$

Note that $\left|\binom{[n]}{d}\right| = \binom{n}{d}$.

For $\vec{x} \in \{0,1\}^n$ and $I = \langle i_1, \ldots, i_d \rangle \in \binom{[n]}{d}$, we let $\vec{x}_{|I} \in \{0,1\}^d$ denote the vector $\langle \vec{x}[i_1], \ldots, \vec{x}[i_d] \rangle$. We often slightly abuse notation when working with $\vec{s} = \vec{x}_{|I}$, and let $\vec{s}[i_j]$ denote the element $x[i_j]$ (rather than the i_jth element in $\vec{s}$). We sometimes overload notation and use $I \in \binom{[n]}{d}$ to denote the lexicographic ordinal of I in $\binom{[n]}{d}$ (which is an integer in $\left[\binom{n}{d}\right]$). We use $I \ni i$ to denote the set of sets $\{I \in \binom{[n]}{d} : i \in I\}$. Namely, all the sets in $\binom{[n]}{d}$ which contain the element i.

2.1 Graded Encoding Schemes

We begin with the definition of a graded encoding scheme, due to Garg, Gentry and Halevi [GGH12]. While their construction is very general, for our purposes a more restricted setting is sufficient as defined below.

Definition 2.1 (τ-Graded Encoding Scheme [GGH12])**.** A τ-encoding scheme for an integer $\tau \in \mathbb{N}$ and ring R, is a collection of sets $\mathcal{S} = \{S_{\vec{v}}^{(\alpha)} \subset \{0,1\}^* : \vec{v} \in \{0,1\}^\tau, \alpha \in R\}$ with the following properties:

1. For every index $\vec{v} \in \{0,1\}^\tau$, the sets $\{S_{\vec{v}}^{(\alpha)} : \alpha \in R\}$ are disjoint, and so they are a partition of the indexed set $S_{\vec{v}} = \bigcup_{\alpha \in R} S_{\vec{v}}^{(\alpha)}$.

2. There are binary operations "+" and "−" such that for all $\vec{v} \in \{0,1\}^\tau$, $\alpha_1, \alpha_2 \in R$ and for all $u_1 \in S_{\vec{v}}^{(\alpha_1)}$, $u_2 \in S_{\vec{v}}^{(\alpha_2)}$:

$$u_1 + u_2 \in S_{\vec{v}}^{(\alpha_1+\alpha_2)} \quad \text{and} \quad u_1 - u_2 \in S_{\vec{v}}^{(\alpha_1-\alpha_2)} ,$$

   where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are addition and subtraction in R.

3. There is an associative binary operation "×" such that for all $\vec{v}_1, \vec{v}_2 \in \{0,1\}^\tau$ such that $\vec{v}_1 + \vec{v}_2 \in \{0,1\}^\tau$, for all $\alpha_1, \alpha_2 \in R$ and for all $u_1 \in S_{\vec{v}_1}^{(\alpha_1)}$, $u_2 \in S_{\vec{v}_2}^{(\alpha_2)}$, it holds that

$$u_1 \times u_2 \in S_{\vec{v}_1+\vec{v}_2}^{(\alpha_1 \cdot \alpha_2)},$$

   where $\alpha_1 \cdot \alpha_2$ is multiplication in R.

In this work, the ring R will always be $\mathbb{Z}_p$ for a prime p.

For the reader who is familiar with [GGH12], we note that the above is the special case of their construction, in which we consider only binary index vectors (in the [GGH12] notation, this corresponds to setting $\kappa = 1$), and we construct our encoding schemes to be *asymmetric* (as will become apparent below when we define our zero-text index $\mathbf{vzt} = \vec{1}$).

Definition 2.2 (Efficient Procedures for a τ-Graded Encoding Scheme [GGH12])**.** We consider τ-graded encoding schemes (see above) where the following procedures are efficiently computable.

- Instance Generation: $\mathsf{InstGen}(1^\lambda, 1^\tau)$ outputs the set of parameters *params*, a description of a τ-Graded Encoding Scheme. (Recall that we only consider Graded Encoding Schemes over the set indices $\{0,1\}^\tau$, with zero testing in the set $S_{\vec{1}}$). In addition, the procedure outputs a subset $evparams \subset params$ that is sufficient for computing addition, multiplication and zero testing[5] (but possibly insufficient for encoding or for randomization).

- Ring Sampler: $\mathsf{samp}(params)$ outputs a "level zero encoding" $a \in S_0^{(\alpha)}$ for a nearly uniform $\alpha \in_R R$.

- Encode and Re-Randomize:[6] $\mathsf{encRand}(params, i, a)$ takes as input an index $i \in \tau$ and $a \in S_0^{(\alpha)}$, and outputs an encoding $u \in S_{\vec{e}_i}^{(\alpha)}$, where the distribution of u is (statistically close to being) only dependent on α and not otherwise dependent of a.

- Addition and Negation: $\mathsf{add}(evparams, u_1, u_2)$ takes $u_1 \in S_{\vec{v}}^{(\alpha_1)}, u_2 \in S_{\vec{v}}^{(\alpha_2)}$, and outputs $w \in S_{\vec{v}}^{(\alpha_1+\alpha_2)}$. (If the two operands are not in the same indexed set, then add returns $\perp$). We often use the notation $u_1 + u_2$ to denote this operation when *evparams* is clear from the context. Similarly, $\mathsf{negate}(evparams, u_1) \in S_{\vec{v}}^{(-\alpha_1)}$.

[5] The "zero testing" parameter **pzt** defined in [GGH12] is a part of *evparams*.

[6] This functionality is not explicitly provided by [GGH12], however it can be obtained by combining their encoding and re-randomization procedures.

- Multiplication: $\mathsf{mult}(evparams, u_1, u_2)$ takes $u_1 \in S_{\vec{v}_1}^{(\alpha_1)}, u_2 \in S_{\vec{v}_2}^{(\alpha_2)}$. If $\vec{v}_1 + \vec{v}_2 \in \{0,1\}^\tau$ (i.e. every coordinate in $\vec{v}_1 + \vec{v}_2$ is at most 1), then mult outputs $w \in S_{\vec{v}_1+\vec{v}_2}^{(\alpha_1 \cdot \alpha_2)}$. Otherwise, mult outputs $\perp$. We often use the notation $u_1 \times u_2$ to denote this operation when *evparams* is clear from the context.
- Zero Test: $\mathsf{isZero}(evparams, u)$ outputs 1 if $u \in S_{\vec{1}}^{(0)}$, and 0 otherwise.

In the [GGH12, CLT13] constructions, encodings are *noisy* and the noise level increases with addition and multiplication operations, so one has to be careful not to go over a specified noise bound. However, the parameters can be set so as to support $O(\tau)$ operations, which are sufficient for our purposes. We therefore ignore noise management throughout this manuscript. An additional subtle issue is that with negligible probability the initial noise may be too big. However this can be avoided by adding rejection sampling to samp and therefore ignored throughout the manuscript as well.

As was done in [BR13], our definition deviates from that of [GGH12]. We define two sets of parameters *params* and *evparams*. While the former will be used by the obfuscator in our construction (and therefore will not be revealed to an external adversary), the latter will be used when evaluating an obfuscated program (and thus will be known to an adversary). When instantiating our definition, the guideline is to make *evparams* minimal so as to give the least amount of information to the adversary. In particular, in the known candidates [GGH12, CLT13], *evparams* only needs to contain the zero-test parameter **pzt** (as well as the global modulus).

2.2 Obfuscation

Definition 2.3 (Virtual Black-Box Obfuscator [BGI+12])**.** Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of polynomial-size circuits, where $\mathcal{C}_n$ is a set of boolean circuits operating on inputs of length n. And let $\mathcal{O}$ be a PPTM algorithm, which takes as input an input length $n \in \mathbb{N}$, a circuit $C \in \mathcal{C}_n$, a security parameter $\lambda \in \mathbb{N}$, and outputs a boolean circuit $\mathcal{O}(C)$ (not necessarily in $\mathcal{C}$).

$\mathcal{O}$ is an *obfuscator* for the circuit family $\mathcal{C}$ if it satisfies:

1. *Preserving Functionality:* For every $n \in \mathbb{N}$, and every $C \in \mathcal{C}_n$, and every $\vec{x} \in \{0,1\}^n$, with all but $negl(\lambda)$ probability over the coins of $\mathcal{O}$:
$$(\mathcal{O}(C, 1^n, \lambda))(\vec{x}) = C(\vec{x})$$
2. *Polynomial Slowdown:* For every $n, \lambda \in \mathbb{N}$ and $C \in \mathcal{C}$, the circuit $\mathcal{O}(C, 1^n, 1^\lambda)$ is of size at most $poly(|C|, n, \lambda)$.
3. *Virtual Black-Box:* For every (non-uniform) polynomial size adversary $\mathcal{A}$, there exists a (non-uniform) polynomial size simulator $\mathcal{S}$, such that for every $n \in \mathbb{N}$ and *for every* $C \in \mathcal{C}_n$:
$$\left| \Pr_{\mathcal{O},\mathcal{A}}[\mathcal{A}(\mathcal{O}(C, 1^n, 1^\lambda)) = 1] - \Pr_{\mathcal{S}}[\mathcal{S}^C(1^{|C|}, 1^n, 1^\lambda) = 1] \right| = negl(\lambda)$$

Remark 2.4. *A stronger notion of functionality, which also appears in the literature, requires that with overwhelming probability the obfuscated circuit is correct on* every input simultaneously*. We use the relaxed requirement that for every input (individually) the obfuscated circuit is correct with overwhelming probability (in both cases the probability is only over the obfuscator's coins). We note that our construction can be modified to achieve the stronger functionality property (by using a ring of sufficiently large size and the union bound).*

3 The Generic Graded Encoding Scheme Model

We would like to prove the security of our construction against *generic adversaries*. To this end, we will use the *generic graded encoding scheme* model, which was previously used in [BR13], and is analogous to the *generic group model* (see Shoup [Sho97] and Maurer [Mau05]). In this model, an algorithm/adversary $\mathcal{A}$ can only interact with the graded encoding scheme via oracle calls to the add, mult, and isZero operations from Definition 2.2. Note that, in particular, we only allow access to the operations that can be run using *evparams*. To the best of our knowledge, non-generic attacks on known schemes, e.g. [GGH12], require use of *params* and cannot be mounted when only *evparams* is given.

We use $\mathcal{G}$ to denote an oracle that answers adversary calls. The oracle operates as follows: for each index $\vec{v} \in \{0,1\}^\tau$, the elements of the indexed set $S_{\vec{v}} = \bigcup_{\alpha \in R} S_{\vec{v}}^{(\alpha)}$ are arbitrary binary strings. The adversary $\mathcal{A}$ can manipulate these strings using oracle calls (via $\mathcal{G}$) to the graded encoding scheme's functionalities. For example, the adversary can use $\mathcal{G}$ to perform an add call: taking strings $s_1 \in S_{\vec{v}}^{(\alpha_1)}, s_2 \in S_{\vec{v}}^{(\alpha_2)}$, encoding indexed ring elements $(\vec{v}, \alpha_1), (\vec{v}, \alpha_2)$ (respectively), and obtaining a string $s \in S_{\vec{v}}^{(\alpha_1+\alpha_2)}$, encoding the indexed ring element $(\vec{v}, (\alpha_1 + \alpha_2))$.

We say that $\mathcal{A}$ is a generic algorithm (or adversary) for a problem on graded encoding schemes (e.g. for computing a moral equivalent of discreet log), if it can accomplish this task with respect to *any* oracle representing a graded encoding scheme, see below.

In the add example above, there may be many strings/encodings in the set $S_{\vec{v}}^{(\alpha_1+\alpha_2)}$. One immediate question is *which* of these elements should be returned by the call to add. In our abstraction, for each $\vec{v} \in \{0,1\}^\tau$ and $\alpha \in R$, $\mathcal{G}$ always uses a *single unique encoding* of the indexed ring element $(\vec{v}, \alpha)$. I.e. the set $S_{\vec{v}}^{\alpha}$ is a singleton. Thus, the representation of items in the graded encoding scheme is given by a map $\sigma(\vec{v}, \alpha)$ from $\vec{v} \in \{0,1\}^\tau$ and $\alpha \in R$, to $\{0,1\}^*$. We restrict our attention to the case where this mapping has polynomial blowup.

Remark 3.1 (Unique versus Randomized Representation)**.** *We note that the known candidates of secure graded encoding schemes [GGH12, CLT13]* do not provide unique encodings*: their encodings are probabilistic. Nonetheless, in the generic graded encoding scheme abstraction we find it helpful to restrict our attention to schemes with unique encodings. For the purposes of proving security against generic adversaries, this makes sense: a generic adversary should work for* any *implementation of the oracle $\mathcal{G}$, and in particular also for an implementation that uses unique encodings.*

Moreover, our perspective is that unique encodings are more "helpful" to an adversary than randomized encodings: a unique encoding gives the adversary the additional power to "automatically" check whether two encodings are of the same indexed ring element (without consulting the oracle). Thus, we prefer to prove security against generic adversaries even for unique representations.

It is important to note that the set of legal encodings may be very sparse within the set of images of σ, and indeed this is the main setting we will consider when we study the generic model. In this case, the only way for $\mathcal{A}$ to obtain a valid representation of any element in any graded set is via calls to the oracle (except with negligible probability). Finally, we note that if oracle calls contain invalid operators (e.g. the input is not an encoding of an element in any graded set, the inputs to add are not in the same graded set, etc.), then the oracle returns $\perp$.

Random Graded Encoding Scheme Oracle. We focus on a particular randomized oracle: the random generic encoding scheme (GES) oracle $\mathcal{RG}$. $\mathcal{RG}$ operates as follows: for each indexed

ring element (with index $\vec{v} \in \{0,1\}^\tau$ and ring element $\sigma \in R$), its encoding is of length $\ell = (|\tau| \cdot \log |R| \cdot poly(\lambda))$ (where $|\tau|$ is the bit representation length of τ). The encoding of each indexed ring element is a uniformly random string of length ℓ. In particular, this implies that the only way that $\mathcal{A}$ can obtain valid encodings is by calls to the oracle $\mathcal{RG}$ (except with negligible probability).

The definition of secure obfuscation in the random GES model is as follows.

Definition 3.2 (Virtual Black-Box in the Random GES Model)**.** Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of circuits and $\mathcal{O}$ a PPTM as in Definition 2.3.

A generic algorithm $\mathcal{O}^{\mathcal{RG}}$ is an obfuscator *in the random generic encoding scheme model*, if it satisfies the functionality and polynomial slowdown properties of Definition 2.3 with respect to $\mathcal{C}$ and to *any* GES oracle $\mathcal{RG}$, but the virtual black-box property is replaced with:

3. *Virtual Black-Box in the Random GES Model:* For every (non-uniform) polynomial size *generic* adversary $\mathcal{A}$, there exists a (non-uniform) generic polynomial size simulator $\mathcal{S}$, such that for every $n \in \mathbb{N}$ and every $C \in \mathcal{C}_n$:

$$\left|\left(\Pr_{\mathcal{RG},\mathcal{O},\mathcal{A}}[\mathcal{A}^{\mathcal{RG}}(\mathcal{O}^{\mathcal{RG}}(C,1^n,1^\lambda))] = 1\right) - \left(\Pr_{\mathcal{RG},\mathcal{S}}[\mathcal{S}^C(1^{|C|},1^n,1^\lambda)] = 1\right)\right| = negl(\lambda)$$

We remark that while it makes sense to allow $\mathcal{S}$ to access the oracle $\mathcal{RG}$, this is in fact not necessary. This is since $\mathcal{RG}$ can be implemented in polynomial time (as described below), and therefore $\mathcal{S}$ can just implement it by itself.

Online Random GES Oracle. In our proof, we will use the property that the oracle $\mathcal{RG}$ can be approximated to within negligible statistical distance by an *online polynomial time process*, which samples the representations on-the-fly. Specifically, the oracle will maintain a table of entries of the form $(\vec{v}, \alpha, \mathsf{label}_{\vec{v},\alpha})$, where $\mathsf{label}_{\vec{v},\alpha} \in \{0,1\}^\ell$ is the representation of $S_{\vec{v}}^{(\alpha)}$ in $\mathcal{RG}$ (the table is initially empty). Every time $\mathcal{RG}$ is called for some functionality, it checks that its operands indeed correspond to an entry in the table, in which case it can retrieve the appropriate $(\vec{v}, \alpha)$ to perform the operation (if the operands are not in the table, $\mathcal{RG}$ returns $\perp$). Whenever $\mathcal{RG}$ needs to return a value $S_{\vec{v}}^{(\alpha)}$, it checks whether $(\vec{v}, \alpha)$ is already in the table, and if so returns the appropriate $\mathsf{label}_{\vec{v},\alpha}$. Otherwise it samples a new uniform label, and inserts a new entry into the table.

When interacting with an adversary that only makes a polynomial number of calls, the online version of $\mathcal{RG}$ is within negligible statistical distance of the offline version (in fact, the statistical distance is exponentially small in λ) . This is because the only case when the online oracle implementation differs from the offline one is when when the adversary guesses a valid label that it has not seen (in the offline setting). This can only occur with exponentially small probability due to the sparsity of the labels. The running time of the online oracle is polynomial in the number of oracle calls.

4 Obfuscating $\mathcal{CNC}^0$

We formally define $\mathcal{CNC}^0$ circuits (conjunctions of $\mathcal{NC}^0_d$ circuits).

Definition 4.1 ($\mathcal{CNC}^0_d$)**.** For an integer d and input length n, a $\mathcal{CNC}^0_d$ circuit C is a predicate on n-bit inputs. It is defined by a set of $\mathcal{NC}^0_d$ subcircuits $(C_1, \ldots, C_m)$, where $m \leq \binom{n}{d}$.

We associate each subcircuit C_I with a set $I \in [d]^n$ of input bits on which C_I operates. Without loss of generality, we always consider the case $m = \binom{n}{d}$, where for every d-tuple $I \in \binom{[n]}{d}$ of distinct input indices, there is a unique subcircuit C_I in C over the variables $\{x_i\}_{i \in I}$.

For every input $\vec{x} \in \{0,1\}^n$:

$$C(\vec{x}) \equiv \bigwedge_{I \in \binom{[n]}{d}} C_I(\vec{x})$$

Observe that any d-CNF is (by definition) a $\mathcal{CNC}^0_d$-circuit (where each sub-circuit computes a d-variable disjunction). We note that the above definition is without loss of generality, because we can always add dummy sub-circuits that output the constant 1 function so that C has a sub-circuit C_I for each $I \in \binom{[n]}{d}$. We can also eliminate multiple sub-circuits on the same I (or son subsets of I), by merging them into a single sub-circuit computing their conjunction (which is still in $\mathcal{CNC}^0_d$).

Remark 4.2. *Unlike many classes of functionalities that were previously studied in the context of obfuscation, even given an accepting input to a $\mathcal{CNC}^0$ circuit C, it might still be hard to recover C using black-box access. This is in contract to, say, point functions [Can97], or even conjunctions [BR13], and it makes obfuscation more challenging.*

An ensemble of conjunctions of $\mathcal{CNC}^0$ circuits is defined in the natural way.

Definition 4.3 (Ensemble of $\mathcal{CNC}^0_d$ Circuits)**.** An ensemble $\mathcal{C} = \{C^{(n)}\}_{n \in \mathbb{N}}$ is a collection of $\mathcal{CNC}^0_d$ circuits, one for each input length.

Remark 4.4. *When we obfuscate a $\mathcal{CNC}^0_d$ circuit C, we allow the obfuscation to expose d (but nothing more). We also allow the obfuscator to run in time $poly(\lambda, n^d)$.*

Our obfuscator for the class of conjunctions is presented in Figure 1. Correctness follows in a straightforward manner as described in the following lemma (the proof is omitted). We note that the error is one sided, it is always the case that if $C(\vec{x}) = 1$ then for the obfuscated program $\mathcal{O}_C(\vec{x}) = 1$ as well.

Lemma 4.5 (Obfuscator Functionality)**.** *Let C be an n-variable $\mathcal{CNC}^0_d$ circuit and let $\mathcal{O}_C = \mathsf{CNCZObf}(1^\lambda, 1^n, 1^d, C)$. Then for every $\vec{x} \in \{0,1\}^n$,*

$$\Pr[\mathcal{O}_C(\vec{x}) \neq \mathcal{C}(\vec{x})] \leq poly(n)/p \ ,$$

where $p = 2^{\Omega(\lambda)}$ is the order of the group in the graded encoding scheme, and the probability is taken over the randomness of CNCZObf*.*

5 Security Against Generic Adversaries

In this section, we prove virtual black-box security for the obfuscator CNCZObf in the random GES model, as per Definition 3.2. The security relies on a new *worst case* complexity theoretic assumption, which we call the *bounded speedup hypothesis* (BSH). This assumption is a "scaled-down" version of the exponential time hypothesis (ETH) [IP99]: whereas the latter asserts that SAT solvers cannot do much better than a brute force search over the space of 2^n solutions, BSH asserts that even solving SAT over smaller solution spaces cannot improve much over brute force search. See Assumption 5.3 below for a formal statement. In Appendix A we discuss the relation

Obfuscator $\mathsf{CNCZObf}$**, on input** $(1^\lambda, 1^n, 1^d, C = (C_1, \ldots, C_m))$ **where** $m = \binom{n}{d}$

1. generate $(params, evparams) \leftarrow \mathsf{InstGen}(1^\lambda, 1^{m+1})$
2. for each $I \in \binom{[n]}{d}$ and $z \in \{0,1\}$: $a_{I,z} \leftarrow \mathsf{samp}(params) \in S_0^{(\alpha_{I,z})}$:
3. for each $i \in [n]$, let I^* be the first set in $\binom{[n]}{d}$ s.t. $i \in I^*$:
   (a) for each $I \in (\binom{[n]}{d} \backslash I^*)$ s.t. $i \in I$, and each $v \in \{0,1\}$: $b_{I,i,v} \leftarrow \mathsf{samp}(params) \in S_0^{(\beta_{I,i,v})}$
   (b) $b'_i \leftarrow samp(params) \in S_0^{(\beta'_i)}$
   $b_{I^*,i,0} \leftarrow b'_i \times \left(\prod_{I \in (\binom{[n]}{d} \backslash I^*): i \in I} b_{I,i,1}\right) \in S_0^{(\beta_{I^*,i,0})}$
   $b_{I^*,i,1} \leftarrow b'_i \times \left(\prod_{I \in (\binom{[n]}{d} \backslash I^*): i \in I} b_{I,i,0}\right) \in S_0^{(\beta_{I^*,i,1})}$
   (c) $c_i \leftarrow \prod_{I \in \binom{[n]}{d}: i \in I} b_{I,i,0} \in S_0^{(\gamma_i)}$, where $\gamma_i = \prod_{I \in \binom{[n]}{d}: i \in I} \beta_{I,i,0}$,
   and note that $\gamma_i = \beta'_i \cdot \prod_{I \in (\binom{[n]}{d} \backslash I^*): i \in I} (\beta_{I,i,0} \cdot \beta_{I,i,1}) = \prod_{I \in \binom{[n]}{d}: i \in I} \beta_{I,i,1}$
4. for each $I \in \binom{[n]}{d}$ and $\vec{s} \in \{0,1\}^d$:
   (a) $d_{I,\vec{s}} \leftarrow (a_{I,C_I(\vec{s})} \cdot \prod_{i \in I} b_{I,i,\vec{s}[i]}) \in S_0^{(\alpha_{I,C_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]})}$
   and denote $\delta_{I,\vec{s}} = (\alpha_{I,C_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]})$
   (b) $r_{I,\vec{s}} \leftarrow \mathsf{samp}(params) \in S_0^{(\rho_{I,\vec{s}})}$
   (c) $s_{I,\vec{s}} \leftarrow (r_{I,\vec{s}} \times d_{I,\vec{s}}) \in S_0^{(\rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}})}$
   (d) $\mathbf{w}_{I,\vec{s}} \leftarrow \mathsf{encRand}(params, I, r_{I,\vec{s}}) \in S_{\vec{e}_I}^{(\rho_{I,\vec{s}})}$
   (e) $\mathbf{u}_{I,\vec{s}} \leftarrow \mathsf{encRand}(params, I, s_{I,\vec{s}}) \in S_{\vec{e}_I}^{(\rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}})}$
5. Letting chk denote the $(m+1)$ group:
   $d_{\mathsf{chk}} \leftarrow \left(\prod_{I \in \binom{[n]}{d}} a_{I,1} \times \prod_{i \in I} c_i\right) \in S_0^{(\delta_{\mathsf{chk}})}$, where $\delta_{\mathsf{chk}} = (\prod_{I \in \binom{[n]}{d}} \alpha_{I,1} \cdot \prod_{i \in [n]} \gamma_i)$
   $r_{\mathsf{chk}} \leftarrow \mathsf{samp}(params) \in S_0^{(\rho_{\mathsf{chk}})}$
   $s_{\mathsf{chk}} \leftarrow r_{\mathsf{chk}} \times d_{\mathsf{chk}} \in S_0^{(\rho_{\mathsf{chk}} \cdot \delta_{\mathsf{chk}})}$
   $\mathbf{w}_{\mathsf{chk}} \leftarrow \mathsf{encRand}(params, \mathsf{chk}, r_{\mathsf{chk}}) \in S_{\vec{e}_{\mathsf{chk}}}^{(\rho_{\mathsf{chk}})}$
   $\mathbf{u}_{\mathsf{chk}} \leftarrow \mathsf{encRand}(params, \mathsf{chk}, s_{\mathsf{chk}}) \in S_{\vec{e}_{\mathsf{chk}}}^{(\rho_{\mathsf{chk}} \cdot \delta_{\mathsf{chk}})}$
6. output the obfuscation: $(evparams, (\mathbf{w}_{\mathsf{chk}}, \mathbf{u}_{\mathsf{chk}}), \{(\mathbf{w}_{I,\vec{s}}, \mathbf{u}_{I\vec{s}})\}_{I \in \binom{[n]}{d}, \vec{s} \in \{0,1\}^d})$

Evaluation, on input $\vec{x} \in \{0,1\}^n$

1. $\mathbf{t} \leftarrow \left(\mathbf{w}_{\mathsf{chk}} \times \prod_{I \in \binom{[n]}{d}} \mathbf{u}_{I,\vec{x}_{|I}}\right) \in S_{(1,1,\ldots,1)}^{\left(\rho_{\mathsf{chk}} \cdot (\prod_{I \in \binom{[n]}{d}} \rho_{I,\vec{x}_{|I}}) \cdot (\prod_{I \in \binom{[n]}{d}} \delta_{I,\vec{x}_{|I}})\right)}$
2. $\mathbf{t}' \leftarrow \left(\mathbf{u}_{\mathsf{chk}} \times \prod_{I \in \binom{[n]}{d}} \mathbf{w}_{I,\vec{x}_{|I}}\right) \in S_{(1,1,\ldots,1)}^{\left(\rho_{\mathsf{chk}} \cdot (\prod_{I \in \binom{[n]}{d}} \rho_{I,\vec{x}_{|I}}) \cdot \delta_{\mathsf{chk}}\right)}$
3. output the bit: $\mathsf{isZero}(evparams, (\mathbf{t} - \mathbf{t}'))$.

Figure 1: Obfuscator for Conjunctions.

between BSH and ETH, and show that the latter implies the former for some parameter range. We find the occurrence of a complexity-theoretic worst-case assumption in a cryptographic proof fairly unusual and intriguing. Whether a proof can be derived without this assumption is left as open problem.

Formally, to conclude that CNCZObf is secure against generic adversaries the following theorem will be proven.

Theorem 5.1. *Under the bounded speedup hypothesis,* CNCZObf *is a virtual black-box obfuscator in the random GES model for the class* $\mathcal{CNC}^0$.

The bounded speedup hypothesis is formally defined as follows.

Definition 5.2. Consider a family of sets $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ such that $X_n \subseteq \{0,1\}^n$. We say that an algorithm $\mathcal{A}$ is a $\mathcal{X}$-3-SAT solver if it solves the 3-SAT problem, restricted to inputs in $\mathcal{X}$. Namely, given a 3-CNF formula $\Phi : \{0,1\}^n \to \{0,1\}$, $\mathcal{A}$ finds whether there exists $x \in X_n$ such that $\Phi(x) = 1$.

Note that a standard self reduction argument shows that if $\mathcal{A}$ is efficient, then the alleged x can also be found efficiently. This is done by restricting each variable in turn to either 0 or 1, and checking if the formula is still satisfiable by an assignment in X_n.

Assumption 5.3 (Bounded Speedup Hypothesis)**.** *There exists a polynomial $p(\cdot)$, such that for any $\mathcal{X}$-3-SAT solver that runs in time $t(\cdot)$, the family of sets $\mathcal{X}$ is of size at most $p(t(\cdot))$.*

In Appendix A we show that a quasi-polynomial version of BSH is in fact implied by ETH. We note that this implies that allowing quasi-polynomial time adversaries and quasi-polynomial time simulators, our proof can be based on ETH.

Assumption 5.4 (quasi-Bounded Speedup Hypothesis (qBSH))**.** *For any $\mathcal{X}$-3-SAT solver that runs in time $t(n)$, the family of sets $\mathcal{X}$ is of size at most $(t(n))^{O(\log(n))}$.*

The remainder of this section is dedicated to proving Theorem 5.1. We follow the methodology of [BR13] and share some of their notation. Our simulator will attempt to execute the adversary $\mathcal{A}$ while simulating the online $\mathcal{RG}$ oracle (see Section 3). In some cases, however, the simulator needs to output labels for unknown ring elements (most notably, the simulated obfuscated program itself). In those cases, $\mathcal{S}$ returns a completely random label, and marks the variable as an unknown in its table. Throughout the execution, $\mathcal{S}$ keeps track of the relations between the unknown variables. This is important because if two variables, whose underlying encoded ring elements are both unknown to the simulator, are supposed to be equal (when the real $\mathcal{RG}$ is used), then they should also be given the same label in the simulation. To this end, we will show that $\mathcal{S}$ can always check for equality of two unknown variables in the obfuscation (or rather, test if their difference is equal to zero) given only oracle access to C. In our construction, the relations between the unknowns are much more complicated than in [BR13] and therefore we need to develop new tools in order to perform the simulation.

For a security parameter $\lambda \in \mathbb{N}$, consider the obfuscation of a $\mathcal{CNC}^0$ circuit C. The obfuscation is generated by choosing parameters for a graded encoding scheme, and then picking ring elements for the various labelings. The output labelings are defined by the underlying ring elements $\{\rho_{I,\vec{s}}, \delta_{I,\vec{s}}\}_{I\in\binom{[n]}{d}\cup\mathsf{chk}}$. *Note that these ring elements are not independently distributed* (though each

of them, on its own, is close to uniformly random in R). In the security proof, we treat these ring elements as variables, since they are unknown to the simulator (all that $\mathcal{S}$ knows is that these variables were sampled in the process of obfuscating a circuit). We use $\vec{\rho}$ to denote the vector containing $\{\rho_{I,\vec{s}}\}_{I\in\binom{[n]}{d}\cup\mathsf{chk}}$, and we call these the ρ random variables. We use $\vec{\delta}$ to denote the vector containing $\{\delta_{I,\vec{s}}\}_{I\in\binom{[n]}{d}\cup\mathsf{chk}}$, and we call these the δ random variables. For a circuit C, we consider the joint distribution of these variables:

Definition 5.5 (Distribution D_C)**.** For a $\mathcal{CNC}^0$ circuit C, we define the distribution D_C over $(\vec{\rho},\vec{\delta})$ to be the distribution of the ρ and δ variables in an obfuscation of C using the obfuscator $\mathsf{CNCZObf}$ in Figure 1.

The Simulator $\mathcal{S}$. For a polynomial-size generic adversary $\mathcal{A}$ (w.l.o.g. we assume that $\mathcal{A}$ is deterministic), the simulator $\mathcal{S}$ simulates $\mathcal{A}$'s operation with an *online* random graded encoding scheme (GES) oracle $\mathcal{RG}$ (see Section 3), i.e. $\mathcal{S}$ answers $\mathcal{A}$'s oracle queries to $\mathcal{RG}$. For a circuit C, we consider the "real" view $\mathcal{A}^{\mathcal{RG}}(\mathsf{CNCZObf}(C))$, where the randomness is over the choice of random GES oracle and the obfuscator's random coins. We also consider a "simulated" view where $\mathcal{S}$ executes $\mathcal{A}$ on a "dummy obfuscation" and answers its oracle queries to $\mathcal{RG}$ (using a strategy specified below). In the simulated view the randomness is over $\mathcal{S}$'s coins.

As outlined above, $\mathcal{S}$ will maintain a table T, where each entry is of the form $(\vec{v},\sigma,\mathsf{label}_{\vec{v},\sigma})$, similarly to the online version of $\mathcal{RG}$ (see Section 3). However, in some cases, instead of setting σ to be an explicit element in the ring R, it sets σ to be an unknown variable, or an arithmetic circuit over R on unknown variables (note that the indexed set $\vec{v}$ is always known).

Initially, the simulator $\mathcal{S}$ creates a "dummy obfuscation". This requires sampling values for $\vec{\rho},\vec{\delta}$. However, $\mathcal{S}$ will mark these values as unknowns, and sample random labels for them, and insert them into the table.[7] By Claim 5.6, the view of the adversary in the simulated setting is statistically indistinguishable from the real setting. This is the only place in the simulation where new unknowns are introduced.

From this point and on, $\mathcal{S}$ needs to answer $\mathcal{A}$'s oracle queries to $\mathcal{RG}$. This is done exactly in the same way as the online implementation of $\mathcal{RG}$. For arithmetic operations, $\mathcal{S}$ can pull the operands from the table T and perform the required operation, either explicitly, if the σ values of the operands are known, or implicitly by creating an arithmetic circuit that combines the arithmetic circuits of the operands. The problem arises in determining the label to be returned as the output of the operation. Recall that if the $(\vec{v},\sigma)$ of the output already appears in T, the same label is to be returned. This means that in order for the simulation to be successful, we must have a way to compare two implicit representations (in the form of arithmetic circuits over unknowns) and determine whether they represent the same value under the distribution D_C. $\mathcal{S}$ needs to do this using only black-box access to C (and the same technique can be used to simulate the isZero operation).

We show how $\mathcal{S}$ can use its *black-box* access to C to compare implicit representations. We use the *functional representation* of the elements encoded in the table entries, as functions of the $(\vec{\rho},\vec{\delta})$ random variables (this method was introduced in [BR13]).

As outlined above, for each entry $t\in T$, by tracking $\mathcal{A}$'s oracle calls, the simulator $\mathcal{S}$ can compute a polynomial-size arithmetic circuit that computes entry t's functional representation f_t.

[7] We note that $\mathcal{S}$ could actually explicitly sample the $\vec{\rho}$ values since they are independent of the circuit C, however it is more convenient for our analysis to leave them undetermined.

The adversary is limited to multi-linear operations over the inputs, so the functional representations have very specific *cross-linear* structure (see below). We use this structure to show that whenever $\mathcal{S}$ needs to check equality of an entry t to 0, $\mathcal{S}$ can determine (given the arithmetic circuit that computes f_t and black-box access to C) whether w.h.p. over $(\vec{\rho}, \vec{\delta}) \sim D_C$, it is the case that $f_t(\vec{\rho}, \vec{\delta}) = 0$. Similarly, $\mathcal{S}$ can use this zero-testing procedure to identify whether two table entries are equal, by comparing their difference function to 0. This is used to determine whether the output of a new oracle call to $\mathcal{RG}$ is already in the table (and if so, to return its label rather than create a new one). Thus, $\mathcal{S}$ can answer all of the adversary's oracle queries, and generate a statistically close view to the real execution.

The remainder of the proof is organized as follows: first, we formalize the above notions of functional representations and their structural properties. We then show that for any table entry t, the simulator can determine with high accuracy and in polynomial time whether or not the functional representation $f_t(\vec{\rho}, \vec{\delta})$ equals 0 w.h.p. when $(\vec{\rho}, \vec{\delta})$ are drawn from D_C. This will allow to complete the simulation as described above. The latter is proven under the bounded speedup hypothesis (Assumption 5.3), which we view as a flavor of the exponential time hypothesis.

5.1 Functional Representations and Structural Properties

We start by exploring the probability space underlying the distribution D_C. As explained above, and can be seen from the description of CNCZObf, the $\vec{\rho}$ variables are uniform and independent. The $\vec{\delta}$ variables, in contrast, are products of more basic variables. We recall that these variables include:

1. the variables $\{\alpha_{I,v}\}_{I \in \binom{[n]}{d}, v \in \{0,1\}}$, which are all uniform and independent in R.
2. The variables $\{\beta_{I,i,v}\}$, for all $i \in [n]$, $I \ni i$, except the lexicographically first I^* for each i, and $v \in \{0,1\}$. These are also uniform and independent (and also independent of the previous variables). We note that the variables $\beta_{I^*,i,v}$ are also defined, but are not independent of the other variables, see below.
3. The variables $\{\beta'_i\}_{i \in [n]}$ that are also uniform and independent in R (and also independent of the previous variables).

Based on these variables, we define $\beta_{I^*,i,v}$, for all i, as follows:

$$\beta_{I^*,i,v} = \beta'_i \cdot \prod_{\substack{I \ni i, \\ I \neq I^*}} \beta_{I,i,1-v} \ .$$

Note that this implies that $\beta_{I^*,i,v}$ are uniform subject to the constraint that for all i,

$$\prod_{I \ni i} \beta_{I,i,0} = \prod_{I \ni i} \beta_{I,i,1} \quad (= \gamma_i) \ .$$

We note the important property that all $\beta_{I,i,v}, \beta'_i, \gamma_i$ variables are completely independent between different i's.

Finally, $\delta_{I,\vec{s}}$ is defined as

$$\delta_{I,\vec{s}} = \alpha_{I,C_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]} \ ,$$

and

$$\delta_{\mathsf{chk}} = \prod_{I \in \binom{[n]}{d}} \alpha_{I,1} \cdot \prod_{i \in [n]} \gamma_i \ .$$

Note that the only variables that depend on the circuit C are $\delta_{I,\vec{s}}$, and in particular this dependence is only on whether the clause C_I is satisfied by $\vec{s}$. We observe that the ring elements encoded in the obfuscation are (w.h.p.) all distinct:

Claim 5.6. *Let $C \in \mathcal{NC}^0$. With all but $negl(\lambda)$ probability over $(\vec{\rho}, \vec{\delta}) \sim D_C$, the encoded ring elements in the obfuscation are all distinct.*

Proof. The encoded ring elements in the obfuscator's output are $\{\rho_{I,\vec{s}}, \rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}}\}_{I \in \binom{[n]}{d} \cup \mathsf{chk}, \vec{s} \in \{0,1\}^d}$. By the properties of the probability space underlying D_C, we observe that with all but $negl(\lambda)$ probability it is the case that $\delta_{I,\vec{s}} \notin \{0,1\}$. Moreover, the $\vec{\rho}$ variables are all uniform and independently random ring elements, and so the probability that the encoded ring elements in the obfuscation are not all distinct is $negl(\lambda)$. □

Building on [BR13], we consider $\mathcal{A}$'s oracle calls, their functional representations, and the structure enforced by the random generic encoding scheme model. We examine $\mathcal{A}$'s oracle calls in sequence. For the $\mathsf{add}, \mathsf{negate}, \mathsf{mult}$ procedures, the indexed ring element in their output is always a multilinear function of the $(\vec{\rho}, \vec{\delta})$ random variables. Thus the table T will only contain multilinear functions of the unknowns $\vec{\rho}, \vec{\delta}$. In fact, these are multilinear functions of $(\vec{\rho}, \vec{\delta})$ with a very specific structure: we call these *cross-linear* functions, see Definition 5.11 and Claim 5.12 below. The difficulty in generating a simulated view is zero testing of a multilinear function under the distribution D_C of its unknowns (namely, the zero-test outcome may depend on the circuit C).

We proceed as follows. Recall that the oracle $\mathcal{RG}$ maintains a table T of all indexed ring elements whose labelings have already been specified (see Section 3). For each element in the table T, we consider its representation as a (cross-linear) function of the $(\vec{\rho}, \vec{\delta})$ random variables. After the obfuscated circuit is specified, the entries in T contain the indexed elements $\{\vec{\rho}_{I,\vec{s}}\}_{I \in \binom{[n]}{d} \cup \mathsf{chk}, \vec{s} \in \{0,1\}^d}$ and $\{\rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}}\}_{I \in \binom{[n]}{d} \cup \mathsf{chk}, \vec{s} \in \{0,1\}^d}$. As $\mathcal{A}$ makes additional oracle calls, new entries are added to the table T. For these new entries, we consider the representations of their indexed ring elements as cross-linear functions of $(\vec{\rho}, \vec{\delta})$. We focus here on calls to the $\mathsf{add}, \mathsf{negate}, \mathsf{mult}, \mathsf{encRand}$ procedures (calls to samp are handled by sampling an element and a labeling, and adding them as an entry to the table with $\tau = 0$).[8] For each such call, the inputs should be labelings that are in the table T: for any other string, the probability that it is a valid labeling is negligible, and the oracle $\mathcal{RG}$ just answers $\perp$ (in both the real execution with a random GES oracle, and in the simulated execution). Otherwise, the inputs are in the table T, and the output may form a new entry in T. The *functional representation* of a table entry is defined as follows.

Definition 5.7 (Functional Representation [BR13])**.** For each entry in $\mathcal{RG}$'s table T, with indexed ring element $(\vec{v}, \sigma) \in \{0,1\}^\tau \times R$ and labeling $s \in \{0,1\}^*$, its *functional representation* $f(\vec{\rho}, \vec{\delta})$ is defined recursively:

[8] We note that in some implementations of Graded Encoding Schemes, e.g. in that of [GGH12], the adversary is only given access to $\mathsf{add}, \mathsf{negate}, \mathsf{mult}$, and so we could consider an even more restricted GES model. All of our results would hold in this restricted model (and no non-generic attacks are known for those schemes).

1. Initially, the only table entries are labelings that appear in the obfuscation. I.e., the entries for the ring elements $\{(\rho_{I,\vec{s}}, (\rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}}))\}_{I \in \binom{[n]}{d}, \vec{s} \in \{0,1\}^d}$, where the variables associated with the index set I are indexed by $\vec{e}_I$. These table entries are all distinct (except with negligible probability), and for each table entry its functional representation is simply $f(\vec{\rho}, \vec{\delta}) = \rho_{I,\vec{s}}$ or $f(\vec{\rho}, \vec{\delta}) = (\rho_{I,\vec{s}} \cdot \delta_{I,\vec{s}})$ (respectively).

2. For subsequent entries that are created on $\mathcal{A}$'s calls to $\mathcal{RG}$, their functional representation is defined recursively. E.g. for an add call, with input labelings s_1 and s_2, if s_1 or s_2 is not in $\mathcal{RG}$'s table T, then that input does not represent a valid labeling, the output is $\perp$ and no new table entry is created. If the inputs are in the table T, let $(\vec{v}_1, \sigma_1)$ and $(\vec{v}_2, \sigma_2)$ be the indexed ring elements that they encode. If $\vec{v}_1 \neq \vec{v}_2$ then again the output is $\perp$ and no new table entry is created. The remaining case is $\vec{v}_1 = \vec{v}_2 = \vec{v}$. In this case, let f_1 and f_2 be the functional representations of the input labeling. If a new table entry is created for the output, then its representation is $(f_1 + f_2)$.

   The cases of $\mathsf{samp}, \mathsf{encRand}, \mathsf{negate}, \mathsf{mult}$ calls are handled similarly.

We remark that for a table entry t, its functional representation is completely independent of the circuit being obfuscated. By definition, the functional representation indeed computes the ring element in a table entry (for any setting of $(\vec{\rho}, \vec{\delta})$). Moreover, for any table entry, $\mathcal{S}$ can compute a polynomial-size arithmetic circuit computing that table entry's functional representation (see [BR13] for the proofs):

Claim 5.8. *For any setting of the initial random variables $(\vec{\rho}, \vec{\delta})$, and for an entry t in the table T containing the indexed ring element $(\vec{v}_t, \sigma_t)$ and with functional functional representation f_t, it is the case that $\sigma_t = f_t(\vec{\rho}, \vec{\delta})$.*

Claim 5.9. *For each entry t in $\mathcal{RG}$'s table T, the simulator $\mathcal{S}$ can compute a polynomial-size arithmetic circuit computing its functional representation $f_t(\vec{\rho}, \vec{\alpha})$.*

As noted above, the functional representations of all table entries are multilinear functions in $(\vec{\rho}, \vec{\delta})$. Moreover, they are multilinear functions with a specific structure, defined as *cross-linear* functions in [BR13]. We proceed to define structural properties of these functions, and then use these to prove security.

Definition 5.10 (cross-linear monomial)**.** A function $h(\vec{\rho})$ is a *cross-linear monomial* of $\vec{\rho}$, or a ρ-monomial, if it is of the form:

$$h(\vec{\delta}) = \eta \cdot (\prod_{I \in S} \rho_{I,y(I)})$$

where $S \subseteq \left(\binom{[n]}{d} \cup \{\mathsf{chk}\}\right)$, $y : \binom{n}{d} \to \{0,1\}^d$ is a partial function defined over S (by notation $y(\mathsf{chk})$ is the empty string), and $\eta \in R$. We also say that h is a cross-linear monomial in the variables $\{\rho_{I,y(I)}\}_{I \in S}$.

Similarly, a function $h(\vec{\delta})$ is a *cross-linear monomial* of $\vec{\delta}$, or a δ-monomial, if it is of the form:

$$h(\vec{\delta}) = \eta \cdot (\prod_{I \in S} \delta_{I,y(I)})$$

where S, y, and η are as above. Similarly to the above, here we say that h is cross-linear in the variables $\{\delta_{I,y(I)}\}_{I \in S}$.

A cross-linear *function* in the variables $\{\delta_{I,y(I)}\}_{I\in S}$ is a sum of cross-linear monomials in the same variables.

Definition 5.11 (cross-linear polynomial [BR13])**.** A function $g(\vec{\rho},\vec{\delta})$ is a *cross-linear term* of $(\vec{\rho},\vec{\delta})$ if it is of the form:

$$g(\vec{\rho},\vec{\delta}) = h_\rho(\vec{\rho})\cdot\left(\sum_{h\in H} h(\vec{\delta})\right)$$

where h_ρ is a ρ-monomial (as per Definition 5.10) w.r.t a set S and assignment y, and for each $h\in H$, h is a δ-monomial over a subset of variables $S'\subseteq S$ and the assignment y.

A function f is a *cross-linear polynomial* of $(\vec{\rho},\vec{\delta})$ if it can be expressed as a sum of cross-linear terms. I.e., it is of the form:

$$f(\vec{\rho},\vec{\delta}) = \sum_{g\in G} g(\vec{\rho},\vec{\delta})$$

where each $g\in G$ is a cross-linear term of $(\vec{\rho},\vec{\delta})$ as above. We assume that the expansion is compact in the sense that the for each term g, the set S (of its ρ-monomial) is distinct. We call this particular expansion the *cross-linear expansion of* f.

It was shown in [BR13] that in the GES model, the functional representations of all table entries are cross-linear polynomials:

Claim 5.12. *For any entry in the table T, its functional representation is a cross-linear polynomial of $(\vec{\rho},\vec{\delta})$.*

5.2 Testing Equality to Zero

For a given table entry t (or difference between table entries), $\mathcal{S}$ needs to test whether its functional representation equals 0. More formally: $\mathcal{S}$ wants to test whether it is the case that when $(\vec{\rho},\vec{\delta})\sim D_C$, with all but negligible probability the output of f_t is zero (we show below that it is always the case that either f_t is always 0, or it is only 0 with negligible probability). We assume that the functional representation $f_t(\vec{\rho},\vec{\delta})$ contains at least one cross-linear term as per Definition 5.11 (otherwise f_t is trivially identically 0 under the distribution D_C).

At this point we deviate from [BR13], and present a new set of tools. To analyze the possible structure of f_t, we introduce the notion of consistent assignments, and consistent monomials:

Definition 5.13 (consistent and full assignments)**.** Let $y:\binom{[n]}{d}\to\{0,1\}^d$ be a partial function (i.e. one that is not necessarily defined on its entire domain). We say that y is *consistent with* $\vec{x}\in\{0,1\}^n$, if for all $I\in\binom{[n]}{d}$ for which $y(I)$ is defined, it holds that $\vec{x}_{|I}=y(I)$. We say that y is *consistent* if it is consistent with some $\vec{x}$. If y is a total function, we say that it is also *full*.

Definition 5.14 (full and consistent ρ-monomials)**.** We say that h is a *full* ρ-monomial if it is a ρ-monomial of total degree $\binom{n}{d}+1$, i.e. of maximal total degree. In this case, for some total function $y:\binom{[n]}{d}\to\{0,1\}^d$, we have $h=\eta\cdot\left((\prod_{I\in\binom{n}{d}}\rho_{I,y(I)})\cdot\rho_{\mathsf{chk}}\right)$. We say that h is the *full ρ-monomial for y*. Further, if y is a *consistent* assignment for some $\vec{x}\in\{0,1\}^n$, as per Definition 5.13, then h is a *full and consistent ρ-monomial* (for the input $\vec{x}$).

Lemma 5.15 below shows that if f_t is *not* a sum of full and consistent ρ-monomials corresponding to accepting inputs for C (each multiplied by a polynomial in δ), then f_t is only 0 with negligible probability over D_C.

The Zero Testing Procedure. The simulator $\mathcal{S}$ begins by identifying ρ-monomials of f_t. This can be done efficiently by Lemma 5.21. If $\mathcal{S}$ finds a non-full or non-consistent monomial, or a full and consistent monomial for $\vec{x}$ s.t. $C(\vec{x}) = 0$, then by Lemma 5.15 it knows that f_t is non-zero (under the distribution D_C). If f *is* a sum of full and consistent ρ-monomials that all correspond to accepting inputs for C (each multiplied by a polynomial in δ), then by Lemma 5.22, $\mathcal{S}$ can test whether f_t gets a 0 or non-zero value when $(\vec{\rho}, \vec{\delta})$ are drawn by D_C.

The only remaining issue is determining whether or not f is a sum of full and consistent ρ-monomials. Lemma 5.21 shows that $\mathcal{S}$ can identify any k ρ-monomials of f in time $poly(k)$. This is not sufficient, however, because f might be a sum of *super polynomially* many full and consistent ρ-monomials. To rule this out, we show that for an efficient adversary $\mathcal{A}$, with all but negligible probability over its coins, all the differences between the functional representations of table entries that it creates can only be sums of a *polynomial* number of full and consistent ρ-monomials. This is where we use the bounded-speedup hypothesis.

We show in Lemma 5.18 that any function which is a sum of super-polynomially many full and consistent ρ-monomials can be used to solve 3-SAT on a super-polynomial set of inputs, which is in contradiction to the BSH (Assumption 5.3). We note that this only requires black-box access to the functional description.

This finalizes the proof of Theorem 5.1. The formal statements and proof of the aforementioned lemmas follows.

Lemma 5.15. *Let C be any $\mathcal{CNC}^0$ circuit. Let f be any cross-linear polynomial of $(\vec{\rho}, \vec{\delta})$ s.t. $f \not\equiv 0$, where $f = \sum_{g \in G} g(\vec{\rho}, \vec{\delta})$, and each function $g \in G$ is a cross-linear term (as in Definition 5.11).*

If there exists $g^* \in G$ s.t. $g^*(\vec{\rho}, \vec{\delta}) = h_\rho(\vec{\rho}) \cdot \left(\sum_{h \in H} h(\vec{\delta})\right)$, where h_ρ is **not** *a full and consistent ρ-monomial for some input $x \in \{0,1\}^n$ s.t. $C(x) = 1$, then:*

$$\Pr_{(\vec{\rho}, \vec{\delta}) \sim D_C} [f(\vec{\rho}, \vec{\delta}) = 0] = negl(\lambda)$$

Proof. Consider the monomial g^* in the lemma statement. Since g^* is either not full and consistent, or is full and consistent with x such that $C(x) = 0$, we can apply either Claim 5.16 or Claim 5.17 below. We get that the variables $\vec{\delta}$ in g^*, when sampled from the distribution D_C, are statistically close to being uniform and independent. We can thus apply the Schwatz-Zippel lemma to argue that with all but $negl(\lambda)$ probability over $\vec{\delta}$, it holds that $\sum_{h \in H} h(\vec{\delta}) \neq 0$. This is because the total degree of $\sum_{h \in H} h(\vec{\delta}) \neq 0$ is at most $\binom{n}{d} + 1$.

Now, let us consider the function f, restricted to an assignment of *all* δ variables according to D_C. This is now a polynomial in the $\vec{\rho}$ variables, with total degree $\binom{n}{d} + 1$. Furthermore, this polynomial is non-zero with all but negligible probability, since the monomial that corresponds to g^* will not zero out. This implies that by Schwartz-Zippel again, that sampling all $\vec{\rho}$ values according to D_C (namely, independently and almost uniformly), $f(\vec{\rho}, \vec{\delta}) \neq 0$ and the lemma follows.

Claim 5.16. *Let $y : \binom{[n]}{d} \to \{0,1\}^d$ be an assignment that is not consistent or not full. Then the variables $\{\delta_{I,y(I)}\}_{I \in \binom{[n]}{d} \cup \{\mathsf{chk}\}}$ in D_C are $negl(\lambda)$-statistically close to uniform and independent.*

Proof. We prove that if $\mathsf{samp}(params)$ samples labelings of uniform elements in $\mathbb{Z}_p^*$ (rather than in $\mathbb{Z}_p$), then the variables stated in the claim are uniform in $\mathbb{Z}_p^*$ and independent. Since the statistical distance from the real setting is $(poly(n)/p)$, the claim follows.

We show two properties:

1. For any assignment y, and for all $i \in [n]$, it holds that $\{\beta_{I,i,y(I)[i]}\}_{I \ni i}$ are uniform and independent. This immediately implies that $\{\beta_{I,i,y(I)[i]}\}_{i \in [n], I \in \binom{[n]}{d}}$ are uniform and independent (since the β values for each i are sampled independently).

2. Consider an assignment y and $j \in [n]$ such that y is either inconsistent or not full on j. Namely, either there exists $J \ni j$ such that J is not in the domain of y or there exist $J_1, J_2 \ni j$ such that $y(J_1)[j] \neq y(J_2)[j]$. Then $\{\beta_{J,j,y(J)[j]}\}_{J \ni j} \cup \{\gamma_j\}$ are all uniform and independent.

Given properties 1 and 2, the claim will follow: Given y as in the claim statement, there must exist j as in property 2. We first consider the set $\{\delta_{I,y(I)}\}_{I \not\ni j}$. This set is uniform and independent based on property 1, since each such $\delta_{I,y(I)}$ can be written a product of variables, where exactly one of these variables is $\beta_{I,i,y(I)[i]}$ (and further, $\beta_{I,i,y(I)[i]}$ does not appear in the product of any of the other $\delta_{I,y(I)}$).

To complete the argument, one needs to show that $\{\delta_{J,y(J)}\}_{J \ni j} \cup \{\delta_{\mathsf{chk}}\}$ are uniform and independent of the former variables. To see this, fix a value for all α, β variables, except for those of the form β'_j or $\beta_{I,j,b}$ (note that γ_j only depends on these variables). Now, each variable in $\{\delta_{J,y(J)}\}_{J \ni j} \cup \{\delta_{\mathsf{chk}}\}$ can be written as a scalar times exactly one variable of the set $\{\beta_{J,j,y(J)[j]})\}_{J \ni j} \cup \{\gamma_j\}$. Specifically $\delta_{J,y(J)}$ is a product of $\beta_{J,j,y(J)[j]}$ with a scalar, and δ_{chk} is a product of γ_j with a scalar. Applying property 2, the independence follows.

Proof of Property 1. Consider some $i \in [n]$ and let I^* be the lexicographically first element in $\{I \in \binom{[n]}{d} : I \ni i\}$. The set $\{\beta_{I,i,y(I)[i]}\}_{I \ni i, I \neq I^*}$ will be uniform and independent since each $\beta_{I,i,b}$ is chosen independently. Further, all of those variables are independent of $\beta_{I^*,i,y(I^*)[i]}$ since the latter contains β'_i which is independent of any of the former β values.

Proof of property 2. From the previous property we have that $\{\beta_{J,j,y(J)[j]}\}_{J \ni j}$ are uniform and independent. It is left to show that γ_j is independent of all of them - assume towards contradiction that this is false. Recall that $\gamma_j = \beta'_j \cdot \prod_{J \neq J^*}(\beta_{J,j,0} \cdot \beta_{J,j,1})$, where J^* is the lexicographically first set that contains j. Further recall that all variables $\beta_{J,j,b}$ for $J \neq J^*$ are sampled uniformly and independently, but for J^* the definition is $\beta_{J^*,j,b} = \beta'_j \cdot \prod_{J \neq J^*} \beta_{J,j,1-b}$, so β'_j only appears as a factor in $\beta_{J^*,j,b}$ (and in γ_j).

If indeed γ_j is not independent of all $\{\beta_{J,j,y(J)[j]}\}_{J \ni j}$, then it must be the case that $y(J^*)$ is defined, since otherwise β'_j will randomize γ_j and make it independent of all other values. Note that γ_j can be written as $\beta_{J^*,j,y(J^*)[j]} \cdot \prod_{J \neq J^*} \beta_{J,j,y(J^*)[j]}$. Since we assume that this product is not independent of $\{\beta_{J,j,y(J)[j]}\}_{J \ni j}$, then all of the factors in the product must appear in the aforementioned set. For counting reasons, and since we know that $\beta_{J^*,j,y(J^*)[j]}$ is indeed in the set, it must also be the case that $\beta_{J,j,y(J^*)[j]}$ is in the set for all J. This in turn implies that $y(J)$ is defined for all J and furthermore that $y(J)[j] = y(J^*)[j]$ for all J.We get a contradiction since we showed that y is consistent and full on j. □

Claim 5.17. *Let $y : \binom{[n]}{d} \to \{0,1\}^d$ be an assignment that is full and consistent with $x \in \{0,1\}^n$ for which $C(x) = 0$. Then the variables $\{\delta_{I,y(I)}\}_{I \in \binom{[n]}{d} \cup \{\delta_{\mathsf{chk}}\}}$ in D_C are $negl(\lambda)$-statistically close to uniform and independent.*

Proof. As in Claim 5.16, we prove that if $\mathsf{samp}(params)$ samples labelings of uniform elements in $\mathbb{Z}_p^*$ (rather than in $\mathbb{Z}_p$), then the variables stated in the claim are uniform in $\mathbb{Z}_p^*$ and independent. Since the statistical distance from the real setting is $poly(n)/p$, the claim follows.

We show that:

- Letting y, x be as in the claim statement, the variables $\{\alpha_{I,C_I(x)}\} \cup \{\prod_I \alpha_{I,1}\}$ are all uniform and independent.

The claim immediately follows since even conditioned on all the β values, each $\delta_{I,y(I)}$ a product that contains $\alpha_{I,C_I(x)}$, and δ_{chk} is a product that contains $\prod_I \alpha_{I,1}$.

The proof of the above property is also fairly straightforward. Since $C(x) = 0$, there exists J such that $\alpha_{J,C_J(x)} = \alpha_{J,0}$. Now, all $\{\alpha_{I,C_I(x)}\}$ are independent by definition (since they are sampled independently), and $\prod_I \alpha_{I,1}$ is independent of all of them since it is a product that contains $\alpha_{J,1}$. □

□

Lemma 5.18. *For $n, \lambda \in \mathbb{N}$, and $X \subseteq \{0,1\}^n$, let f be any cross-linear function of $(\vec{\rho}, \vec{\delta})$ s.t.*

$$f = \sum_{x \in X} h_x(\vec{\rho}) \cdot g_x(\vec{\delta})$$

where $\forall x \in X$, h_x is a full ρ-monomial which is consistent with *x, and $g_x \not\equiv 0$.*

Then there exists a polynomial-size circuit $\mathcal{A}^f$ s.t. for any circuit $B \in \mathcal{NC}^0$ on n-bit inputs, the circuit $\mathcal{A}^f$ on input B decides whether there exist $x \in X$ that satisfies B. I.e.:

$$\forall B \in \mathcal{NC}^0 : \Pr_{\mathcal{A}}[\mathcal{A}(B) = \mathbf{1}_{\exists x \in X : B(x)=1}] \geq 1 - negl(\lambda)$$

Proof. Each $h_x(\vec{\rho})$ is a full ρ-monomial consistent with x, and so:

$$\forall x \in X : h_x(\vec{\rho}) = \prod_{I \in \left(\binom{[n]}{d} \cup \{\mathsf{chk}\}\right)} \vec{\rho}_{I,x_{|I}}$$

The algorithm $\mathcal{A}^f$ picks $(\vec{\rho}, \vec{\delta})$ from a distribution F_B, which depends on the circuit B (but is *not* the distribution D_B that these variables take in the obfuscation). $\mathcal{A}^f$ outputs 1 iff $f(\vec{\rho}, \vec{\delta}) \neq 0$. The distribution F_B is obtained as follows: each variable in $\vec{\delta}$ is chosen uniformly and random from the ring. For each variable $\rho_{I,\vec{s}}$, if $B_I(\vec{s}) = 1$ then it is chosen uniformly at random, but if $B_I(\vec{s}) = 0$ then it is always fixed to 0. The variable $\vec{\rho}_{\mathsf{chk}}$ is uniformly random and independent. The lemma follows from Claims 5.19 and 5.20 below.

Claim 5.19. *If $\forall x \in X, B(x) = 0$, then $\Pr_{\vec{\rho},\vec{\delta}}[f(\vec{\rho}, \vec{\delta}) = 0] = 1$*

Proof. Since $\forall x \in X, B(x) = 0$, we conclude that $\forall x \in X, \exists I \in \binom{[n]}{d} : B_I(x_{|I}) = 0$. By the construction of F_B, we conclude that $\forall x \in X, \exists I \in \binom{[n]}{d} : \rho_{I,x_{|I}} = 0$. Thus $\forall x \in X$, the function $h_x(\vec{\rho})$, which is a product of all of the $\rho_{I,x_{|I}}$ variables, outputs 0. We conclude that $f(\vec{\rho}, \vec{\delta}) = \sum_{x \in X} h_x(\vec{\rho}) \cdot g_x(\vec{\delta})$ is a sum of 0's, and so when f's input is drawn from D_F, f's output is always 0. □

Claim 5.20. *If $\exists x^* \in X : B(x^*) = 1$, then $\Pr_{\vec{\rho},\vec{\delta}}[f(\vec{\rho}, \vec{\delta}) = 0] = negl(\lambda)$.*

Proof. Set all variables in $\vec{\rho}$ that take value 0 in D_F to 0. This gives restricted functions: f' derived from f, and h'_x derived from h_x (for each $x \in X$). Choosing the input $(\vec{\rho}, \vec{\delta}) \sim D_F$ for f is equivalent to choosing a uniformly random input for f'.

For the promised $x^* \in X : B(x^*) = 1$, we get that h'_x is a non-zero polynomial of total degree at most $(\binom{n}{d} + 1)$. Thus f' is a non-zero polynomial of total degree $O(\binom{n}{d} + 1)$, and we are choosing a uniformly random input for it. By Schwartz-Zippel, the probability that f' outputs 0 is $negl(\lambda)$. □

□

Lemma 5.21. *There exists an algorithm that takes an arithmetic circuit computing a cross-linear polynomial*

$$f(\vec{\rho}, \vec{\delta}) = \sum_{k \in [K]} h_k(\vec{\rho}) \cdot (\sum_{h \in H_k} h(\vec{\delta})) \ ,$$

finds, with all but negligible probability, an arithmetic circuit for $h_{k^*}(\vec{\rho}) \cdot (\sum_{h \in H_{k^*}} h(\vec{\delta}))$ for some $k^* \in [K]$, and furthermore finds the assignment $y_{k^*} : (\binom{[n]}{d} \cup \mathsf{chk}) \to \{0,1\}^d$ that is associated with the monomial h_{k^*}.

Proof. First, observe that it is easy to perform zero-testing for functions like f using Schwartz-Zippel (because f is a low-degree function of the underlying ρ, α, β variables, which are drawn uniformly at random). Our algorithm will start by finding a minimal set of ρ variables such that if all other ρ variables are set to 0, the function f remains non-zero. This is easy to do by setting one variable at a time to zero. Note that we need a *minimal* set, i.e. one that cannot be decreased further, and *not a minimum* set. From the structure of f, such a set must correspond to the set of variables in some monomial h_{k^*} . Furthermore, given this set of variables, the assignment y_{k^*} is immediately derived. More explicitly, for all $\rho_{I,\vec{s}}$ in the set, we define $y_{k^*}(I) = \vec{s}$. Since f is cross-linear, there can be no collisions in this definition.

Now, consider the arithmetic circuit obtained from f by fixing the value of all ρ variables, except the aforementioned set, to 0. This circuit computes the function $h_{k^*}(\vec{\rho}) \cdot (\sum_{h \in H_{k^*}} h(\vec{\delta}))$ as required. □

Lemma 5.22. *Let C be any $\mathcal{CNC}^0$ circuit and let let $O \in \mathcal{CNC}^0$ be the circuit for the constant function 1 (the function that accepts any input). Let f be a cross-linear polynomial of the form:*

$$f(\vec{\rho}, \vec{\delta}) = \sum_{x \in X} h_x(\vec{\rho}) \cdot g_x(\vec{\delta}) \ ,$$

where h_x, g_x are consistent with x, and $\forall x \in X, C(x) = 1$. Then:

$$\Pr_{(\vec{\rho},\vec{\delta}) \sim D_C}[f(\vec{\rho}, \vec{\delta}) = 0] = \Pr_{(\vec{\rho},\vec{\delta}) \sim D_O}[f(\vec{\rho}, \vec{\delta}) = 0]$$

Moreover, when $(\vec{\rho}, \vec{\delta}) \sim D_C, D_O$, either it is always *the case that $f(\vec{\rho}, \vec{\delta})$ equals 0, or it only equals 0 with $negl(\lambda)$ probability (and these cases can be distinguished efficienctly).*

Proof. Recall that for any circuit C, in the distribution D_C the variables $\vec{\rho}$ are uniform and independent, whereas the variables $\vec{\delta}$ are determined by the underlying random variables $\{\beta_{I,i,v}, \alpha_{I,v}\}$, which are sampled independently of the circuit C. Recall further that $\delta_{\mathsf{chk}} = \prod_I \alpha_{I,1} \cdot \prod_{i,I} \beta_{I,i,1}$, namely δ_{chk} is also independent of the circuit C.

In contrast, for all $I, \vec{s}$, $\delta_{I,\vec{s}} = \alpha_{I,C_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]}$. I.e. these variables do depend on the circuit C and differ between D_C and D_O. However, since f is a sum of (products of) monomials that are consistent with accepting inputs of C, it follows that all of the δ variables in f are consistent with an accepting input x: i.e. they are of the form $\delta_{I,\vec{s}}$, where there exists an input $x \in X$ s.t. $C(x) = 1$ and $x_{|I} = \vec{s}$, so it must be the case that $C_I(\vec{s}) = 1$. In other words, for all δ values in f, it holds that

$$\delta_{I,\vec{s}} = \alpha_{I,C_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]} = \alpha_{I,1} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]} = \alpha_{I,O_I(\vec{s})} \cdot \prod_{i \in I} \beta_{I,i,\vec{s}[i]} \ ,$$

Thus, the joint distributions of the δ variables in D_C and D_O are identical. The Lemma follows because, viewing $f(\vec{\rho}, \vec{\delta})$ as a function of the ρ, α, β variables (the same function when $(\vec{\rho}, \vec{\delta})$ are drawn from D_C or from D_O), either f is always 0, or it is only 0 with negligible probability (by Schwartz-Zippel, since f is a low-degree function of the uniformly random ρ, α, β). □

Acknowledgments. We thank Shafi Goldwasser for helpful and insightful comments, and Moni Naor for bring to our attention the connections between the Bounded Speedup Hypothesis and parameterized complexity theory. We are grateful to Ryan Williams for illuminating conversations about the Bounded Speedup Hypothesis, and in particular for pointing out the proof that qBSH is implied by ETH [Wil13]. We also thank Amir Yehudayoff for helpful discussions on arithmetic circuit complexity.

References

[AW07] Ben Adida and Douglas Wikström. How to shuffle in public. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 2007.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *CRYPTO*, pages 520–537, 2010.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012. Preliminary version in CRYPTO 2001.

[BR13] Zvika Brakerski and Guy Rothblum. Obfuscating conjunctions, 2013. To appear in CRYPTO 2013.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *IACR Cryptology ePrint Archive*, 2002:80, 2002.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, pages 455–469, 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT*, pages 489–508, 2008.

[CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. Cryptology ePrint Archive, Report 2013/183, 2013. To appear in CRYPTO 2013.

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In *STOC*, pages 131–140, 1998.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, pages 72–89, 2010.

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 654–663. ACM, 2005.

[FG06] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. *IACR Cryptology ePrint Archive*, 2012:610, 2012. To appear in EUROCRYPT 2013.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. Cryptology ePrint Archive, Report 2013/451, 2013. To appear in FOCS 2013.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.

[HMLS10] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. *J. Cryptology*, 23(1):121–168, 2010.

[HRSV11] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-sat. In *IEEE Conference on Computational Complexity*, pages 237–240. IEEE Computer Society, 1999.

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT*, pages 20–39, 2004.

[Mau05] Ueli . Maurer. Abstract models of computation in cryptography. In *IMA Int. Conf.*, pages 1–12, 2005.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In *TCC*, pages 579–598, 2013.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 523–532. ACM, 2005.

[Wil13] Ryan Williams. Personal communication, 2013.

A The Bounded Speedup Hypothesis

In this section we investigate the relation between BSH and ETH. We use an insight of Williams [Wil13] to show that qBSH (quasi-polynomial BSH) is in fact implied by ETH. We also discuss a connection with parameterized complexity. We start by restating BSH, qBSH, ETH using similar terminology.

Definition 5.2 ($\mathcal{X}$-3-SAT solver, restated)**.** Consider a family of sets $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ such that $X_n \subseteq \{0,1\}^n$. We say that an algorithm $\mathcal{A}$ is a $\mathcal{X}$-3-SAT solver if it solves the 3-SAT problem, restricted to inputs in $\mathcal{X}$. Namely, given a 3-CNF formula $\Phi : \{0,1\}^n \to \{0,1\}$, $\mathcal{A}$ finds whether there exists $x \in X_n$ such that $\Phi(x) = 1$.

Assumption 5.3 (BSH, restated)**.** *There exists a polynomial $p(\cdot)$, such that for any $\mathcal{X}$-3-SAT solver that runs in time $t(\cdot)$, the family of sets $\mathcal{X}$ is of size at most $p(t(\cdot))$.*

The quasi-BSH (qBSH), is a variant where the polynomial $p(\cdot)$ is allowed to have degree $\log(n)$.

Assumption 5.4 (qBSH restated)**.** *For any $\mathcal{X}$-3-SAT solver that runs in time $t(n)$, the family of sets $\mathcal{X}$ is of size at most $(t(n))^{O(\log(n))}$.*

We proceed with the exponential time hypothesis, presented by Impagliazzo and Paturi [IP99]. For our purposes, we use the weaker notion of ETH (the strong version makes claims about kCNF-SAT for all k).

Assumption A.1 (The Exponential Time Hypothesis)**.** *There is no $2^{o(n)}$-time $\{0,1\}^n$-3-SAT solver.*

We further consider the following scaled down version of the ETH, which is not stronger than the original.

Assumption A.2 (Scaled-Down ETH)**.** *There is no $t(n)$-time solver for $\omega(\log(t(n)))$-variable 3-SAT.*

Assumption A.1 implies Assumption A.2 by a scaling argument. The Scalded-Down ETH, in turn, is more similar to the BSH: we can view a solver for $\omega(\log(n))$-variable 3SAT as an $\mathcal{X}$ solver where $\mathcal{X}$ covers all possible assignments of the first $\omega(\log(n))$ variables (or rather, any subset of $\omega(\log(n))$ variables). The only difference from BSH thus is that the family $\mathcal{X}$ is "structured" in the sense of being concentrated on a small set of coordinates.

In fact, when we consider non-uniform solvers, the ETH implies qBSH (note that the scaling down above also applies to non-uniform solvers):

Lemma A.3 ([Wil13])**.** *For non-uniform solvers, qBSH is implied by Scaled-Down ETH for (and therefore, qBSH is implied by ETH for non-uniform circuits).*

Proof. We start by recalling a learning theoretic argument. Given a family $\mathcal{X}$, we can consider the strings in X_n as functions in $\{0,1\}^k \to \{0,1\}$, where $k = \lceil \log(n) \rceil$ (the $(2^k - n)$ last bits can be set to zero). Recall that by Sauer's Lemma, the VC dimension of X_n (as a set of boolean functions) is at least $\mathsf{dim} = \Omega(\log(|X_n|)/k)$. This means that there exists a set of dim coordinates that is shuttered by X_n. Namely, such that there exists 2^{dim} functions in X_n that cover all possible assignments to this set of coordinates.

It follows, therefore, that if there exists a $\mathcal{X}$-3-SAT solver $\mathcal{A}$ that runs in time $t(n)$ with $|X_n| = (t(n))^{\omega(log(n))}$, then there is a set of

$$\Omega(\log(|X_n|)/\lceil \log(n) \rceil) = \omega(\log(t(n)))$$

coordinates that is shuttered by X_n (namely, they take all possible assignments when ranging over X_n). Mapping those to the variables of a scaled-down formula, yields a (non-uniform) solver for the Scaled-Down ETH (the coordinates to map to are specified by non-uniform advice). □

Connections with Parameterized Complexity. The bounded-speedup hypothesis implies that 3-SAT is hard even when restricted to super-polynomial-size subsets of inputs. One natural set to consider is the inputs of weight k, where k is super-constant or logarithmic (and weight is the L_1 norm, or the number of 1's in the assignment). This is a super-polynomial set, with a very short implicit description.

The problems of resolving SAT and 3-SAT on inputs of weight k, in time that is $f(k) \cdot poly(n)$ (for any function f, even super-exponential in k), has been studied in the context of parameterized complexity. "Weighted" 3-SAT is known to be complete for the class $W[1]$, and is not generally believed to be tractable. See e.g. [FG06].

This study shows an obstacle to finding a 3-SAT algorithm with significant speedup for *every* set $\mathcal{X}$ of inputs: in particular, doing so for the set of low-weight inputs seems hard. Still, we emphasize the Bounded Speedup Hypothesis asks for more: it says that there *exists no set* with super-polynomial speedup. Reducing this hypothesis to a more well studied worst-case problem is an intriguing direction for future work.