Protecting obfuscation against arithmetic attacks

Eric Miles
UCLA
enmiles@cs.ucla.edu

Amit Sahai*
UCLA
sahai@cs.ucla.edu

Mor Weiss
Technion
morw@cs.technion.ac.il

March 2, 2015

Abstract

Recently, the work of Garg et al. (FOCS 2013) gave the first candidate general-purpose obfuscator. This construction is built upon multilinear maps, also called a graded encoding scheme. Several subsequent works have shown that variants of this obfuscator achieves the highest notion of security (VBB security) against "purely algebraic" attacks, namely attacks that respect the restrictions of the graded encoding scheme. While important, the scope of these works is somewhat limited due to the strong restrictions imposed on the adversary. Further, recent attacks on multilinear map candidates have highlighted the inadequacies of existing security models.

We propose and analyze another variant of the Garg et al. obfuscator in a setting that imposes fewer restrictions on the adversary, which we call the arithmetic setting. This setting captures a broader class of algebraic attack scenarios than considered in previous works. Most notably, it allows for unlimited additions across different "levels" of the encoding, which is allowed by the underlying multilinear maps, but not captured in previous graded encoding models. In this setting, we present two results:

- First, in the arithmetic setting where the adversary is limited to creating only multilinear polynomials, we obtain an unconditional proof of VBB security.
- Second, in the arithmetic setting where the adversary can create polynomials of arbitrary degree, we prove VBB security under a new, natural assumption. We also give evidence that any unconditional proof of VBB security in this model would entail proving the algebraic analog of P $\neq$ NP.

*Supported in part by a DARPA/ONR PROCEED award, NSF grants 1228984, 1136174, 1118096, 1065276,0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

1 Introduction

The goal of general-purpose program obfuscation is to make programs "unintelligible", while preserving their functionality. This field of research, first formalized in the works of Barak et al. [BGI+01] and Hada [Had00], is exciting due to its many possible applications (see the introduction of [AGIS14] for a comprehensive list). Of particular interest to us is the recent work of Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b], which gave the first candidate construction of a general purpose obfuscator.

The heart of the [GGH+13b] construction is an obfuscator for log-depth circuits (NC^1), which can be represented by permutation matrix branching programs [Bar89]. This obfuscator assumes the existence of a graded encoding scheme whose details we review below; candidate implementations of such a scheme can be obtained from the candidate multilinear map constructions of Garg, Gentry, and Halevi [GGH13a] and Coron, Lepoint and Tibouchi [CLT13]. At a high-level, the obfuscator proceeds by randomizing the matrix branching program using Kilian's technique [Kil88], and then encoding each element of each matrix using the graded encoding scheme. (There is a final step which transforms an obfuscator for NC^1 into an obfuscator for all poly-size circuits, but this will not be our focus here.)

The graded encoding scheme imposes some restrictions on the ways in which encoded matrix elements may be added and multiplied. These restrictions are motivated by corresponding restrictions from candidate multilinear map constructions [GGH13a, CLT13]; it is conjectured that in these latter constructions, operations that violate the restrictions do not reveal "useful" information about the underlying encoded elements. Thus an important question, posed by [GGH+13b], is whether their NC^1-obfuscator can be proven secure against "purely algebraic" adversaries who are required to obey the restrictions of the graded encoding scheme.

This question has been addressed by several recent works, which give constructions achieving Virtual Black Box (VBB) security against specific classes of possible attacks. Roughly speaking, VBB guarantees that the obfuscated program reveals nothing other than its input-output behavior, and is the strongest meaningful theoretical notion of obfuscation security. Brakerski and Rothblum [BR14b] show that a variant of the [GGH+13b] obfuscator achieves VBB security against purely algebraic attacks, under a new assumption known as the Bounded Speedup Hypothesis. Barak et al. [BGK+14] remove the need for this assumption, and show unconditionally that another variant of the [GGH+13b] obfuscator achieves VBB security against purely algebraic attacks. Finally, Ananth et al. [AGIS14] give a more efficient variant of the [BGK+14] obfuscator and show that it also achieves VBB security against purely algebraic attacks.

The main drawback of the works [BR14b, BGK+14, AGIS14] is that the algebraic restrictions imposed by the graded encoding scheme are quite strong. Though these restrictions are motivated by current multilinear map constructions (and in particular rely on the assumption that operations violating the restrictions reveal no "useful" information on the encoded elements), it is vastly preferable to prove security while allowing as broad a set of algebraic operations as possible. This is all the more imperative in light of recent attacks on multilinear map constructions. (See Section 1.2 for context on these attacks.)

In particular, if we delve deeper into existing constructions of multilinear maps [GGH13a, CLT13], there are several *purely algebraic operations* that an adversary can perform in these existing multilinear map constructions that do not correspond to algebraic operations in the generic graded encoding model. Most notably, in [GGH13a, CLT13], all encodings, regardless their "level" of the graded algebra, are represented as polynomials in a single ring. Thus, an adversary can add encodings at disparate levels using an arithmetic operation, and yet this is not captured in the generic models considered in [BR14b, BGK+14, AGIS14]. If we are able to deal with broader classes of algebraic attacks, this not only furthers our understanding of implementations using current multilinear map constructions, but also allows for greater compatibility with potential future constructions. For example, in future multilinear maps constructions, it may be possible that elements can be zero-tested at any level of the encoding. Our new models allow this, while in all other works the adversary is not permitted this flexibility.

Interestingly, and perhaps surprisingly, we show that the loosening of these restrictions is related to fundamental questions in arithmetic circuit complexity. Specifically, our results below give evidence that proving VBB security in the most general type of graded encoding scheme would imply the algebraic analog of P $\neq$ NP, namely VP $\neq$ VNP.

1.1 Our results

We give an obfuscator for NC^1 circuits, and show that it achieves VBB security against *a broader class of algebraic attacks than considered in all previous works*. The broader class of attacks that we consider, which we call *arithmetic attacks*, is directly inspired by models from arithmetic complexity theory, and by actual *attack scenarios* that could arise using current multilinear map constructions.

To explain our results, we first describe the restrictions imposed by the graded encoding scheme used in [BR14b, BGK+14, AGIS14]. Throughout this section, we let $\mathcal{R}$ denote a commutative ring and $\mathbb{U}$ denote a universe set.

Definition 1.1 (Fully restricted graded encoding scheme)**.** A *fully-restricted graded encoding scheme* consists of a set of basic elements $\{(r_i, S_i)\}_i$ where $r_i \in \mathcal{R}$ and $S_i \subseteq \mathbb{U}$ for each i; three operations $+$, $-$, and $\times$; and a predicate IsZero defined as follows.

- For (r, S) and (r', S), we define $(r, S) + (r', S) := (r + r', S)$ and $(r, S) - (r', S) := (r - r', S)$.
- For (r, S) and (r', S') where $S \cap S' = \emptyset$, we define $(r, S) \times (r', S') := (r \times r', S \cup S')$.
- For (r, S) where $S = \mathbb{U}$, we define $\mathsf{IsZero}((r, S)) :=$ True iff r is $\mathcal{R}$'s zero element.

A fully-restricted graded encoding scheme imposes three restrictions. First, two elements can be added or subtracted only if they have the same "index-set" S. Second, two elements can be multiplied only if their index-sets are disjoint. Third, only elements whose index-set is the universe $\mathbb{U}$ can be zero-tested.

To describe our two less-restricted graded encoding schemes, we use the following notion of a valid polynomial.

Definition 1.2. Let $\{(r_i, S_i)\}_i$ be a set of elements where $r_i \in \mathcal{R}$ and $S_i \subseteq \mathbb{U}$ for each i. We say that a polynomial over the set $\{(r_i, S_i)\}_i$ is *valid* iff it is multilinear and, for every non-zero monomial and every $(r_i, S_i), (r_j, S_j)$ that appear in the monomial, $S_i \cap S_j = \emptyset$.

In our first new scheme, we only require that pairs of basic elements with intersecting index sets are never multiplied. In particular, additions, subtractions and zero-testing are always allowed. This corresponds to allowing any operation that results in a valid polynomial.

Definition 1.3 (Multiplication-restricted graded encoding scheme)**.** A *multiplication-restricted graded encoding scheme* consists of a set of basic elements $\{(r_i, S_i)\}_i$ and arithmetic circuits defined over them. The operations $+$, $-$, $\times$, and IsZero are defined as follows. Let e_1 and e_2 be any arithmetic circuits over the basic elements (where each basic element is a circuit of size 1).

- $e_1 + e_2$ and $e_1 - e_2$ are the formal arithmetic circuits computing those expressions.

- For e_1 and e_2 such that $e_1 \times e_2$ computes a valid polynomial, $e_1 \times e_2$ is the formal arithmetic circuit computing that expression.

- For e_1 that computes a valid polynomial, $\mathsf{IsZero}(e_1) :=$ True iff evaluating e_1 on the basic elements produces $0 \in \mathcal{R}$.

In our second new scheme, we allow all arithmetic operations (even those resulting in invalid polynomials), but with the caveat that any invalid polynomial is always classified as "non-zero".

Definition 1.4 (Unrestricted graded encoding scheme)**.** An *unrestricted graded encoding scheme* consists of a set of basic elements $\{(r_i, S_i)\}_i$ and arithmetic circuits defined over them. The operations $+$, $-$, $\times$, and IsZero are defined as follows. Let e_1 and e_2 be any arithmetic circuits over the basic elements (where each basic element is a circuit of size 1).

- $e_1 + e_2$, $e_1 - e_2$, and $e_1 \times e_2$ are the formal arithmetic circuits computing those expressions.

- For any e_1, we define $\mathsf{IsZero}(e_1) :=$ True iff e_1 computes a valid polynomial and evaluating e_1 on the basic elements produces $0 \in \mathcal{R}$.

Remark 1.5. Another sensible criterion for evaluating IsZero, inspired by current implementations of multilinear maps [GGH13a, CLT13], is the following: a valid element e is 0 iff it evaluates to 0 and further each of its monomials has index set $\mathbb{U}$ (where the index set of monomial $(r_1, S_1) \cdots (r_m, S_m)$ is $\bigcup_{i \leq m} S_i$). Our results below hold for this variant as well.

We now state our main theorems. We defer until Section 2 the formal definition of VBB security and an "ideal" graded encoding scheme (the latter is simply a way of formalizing an adversary that is restricted to the defined set of arithmetic operations).

Our first main theorem shows that in the multiplication-restricted setting, we can achieve the strongest possible notion of security.

Theorem 1.6 (VBB in the multiplication-restricted ideal graded encoding scheme)**.** *For any multiplication-restricted ideal graded encoding scheme, there exists an efficient obfuscator achieving VBB security against all polynomial-time adversaries.*

Our second main theorem shows that, in the unrestricted setting, we can achieve VBB security under a worst-case assumption that is inspired by the Bounded Speedup Hypothesis of [BR14a, BR14b].[1] We defer the details of this hypothesis until Section 5.1, but it corresponds to a parameterized version of the Bounded Speedup Hypothesis that replaces 3SAT with (the decision version of) Max-2-SAT. In particular, note that the running time of the simulator depends on the choice of $p(\cdot)$ in the assumption.

Theorem 1.7 (VBB in the unrestricted ideal graded encoding scheme)**.** *Assume the p-Bounded-Speedup-Hypothesis. Then for any unrestricted ideal graded encoding scheme, there is an efficient obfuscator that achieves VBB security against all time-t adversaries with a time-$(p(t^{O(1)}))^{O(1)}$ simulator.*

As mentioned above, we give evidence that proving an unconditional version of Theorem 1.7 would entail proving VP $\neq$ VNP; see Remark 5.4 in Section 5.

[1] We were informed by Guy Rothblum (personal communication) that Uri Feige has shown that the BSH is false, using a SAT-solver based attack. This is not directly applicable to our assumption, because we use a different NP-complete problem, and indeed our construction can be made to work with other NP-complete problems. Still, in light of the BSH's status, we choose to give a parameterized assumption.

Finally, we note that another active area of research is to show that the [GGH+13b] obfuscator achieves a different notion of security, known as *indistinguishability obfuscation* (iO), against all efficient adversaries. (VBB security against all efficient adversaries is known to be impossible in general [BGI+01].) As a corollary of Theorem 1.7, we can show that our obfuscator unconditionally achieves iO security in the unrestricted setting.

Corollary 1.8 (iO in the unrestricted ideal graded encoding scheme)**.** *For any unrestricted ideal graded encoding scheme, there exists an efficient obfuscator achieving iO security against all polynomial-time adversaries.*

1.2 Our results in the context of recent attacks

After our work first appeared online, a surprising new attack against the [CLT13] multilinear map candidate was shown by Cheon, Han, Lee, Ryu and Stehlé [CHL+14], and was quickly extended by other researchers [GHMS14, BWZ14, CLT14]. These attacks demonstrate a gap between operations permitted in existing generic models (including ours) and those allowed in reality. In particular, the attacks crucially use the fact that the real zero-test returns an element in the ring of encodings, rather than just a bit.

These new attacks only serve to reinforce the need for developing new models that capture a broad class of attack scenarios. With this in mind, we stress the importance of being proactive in modeling which operations the adversary is permitted to use. Though (for example) adding encodings across different levels and zero-testing at low levels has not yet led to concrete attacks, these are very natural operations whose impact on security is well deserving of study.

We also note that in a subsequent work [BMSZ15], one of our new technical contributions was used to obfuscate a limited but natural class of functions in a new model that *does* capture the recent attacks. Specifically, that work crucially uses our strong straddling sets (Construction 2.6).

1.3 Techniques

Our obfuscator in Theorems 1.6 and 1.7 is almost identical to that of [AGIS14], with the primary difference being that we make use of a stronger notion of "straddling sets" than the ones proposed in [BGK+14]. Before outlining our proofs we review this construction. (In fact, the construction is more involved than described here, but this version will suffice for explaining our techniques.)

Obfuscating the branching program. For a given NC^1 function $f : \{0,1\}^\ell \to \{0,1\}$, we first construct an oblivious matrix branching program BP with length $n = \mathsf{poly}(\ell)$ and width $w = \mathsf{poly}(\ell)$ over the field $\mathbb{Z}_p$ for a prime $p = 2^{\Omega(\ell)}$. BP consists of $2n$ non-singular $w \times w$ matrices over $\mathbb{Z}_p$, denoted $\{B_{i,b} \mid i \in [n], b \in \{0,1\}\}$, and a function $\mathsf{inp} : [n] \to [\ell]$ specifying which input bit is read in each layer. (Crucially, the function inp depends only on ℓ and n, and not on f.) For a given input $x \in \{0,1\}^\ell$, let $B_x := \prod_{i=1}^n B_{i,x_{\mathsf{inp}(i)}}$ be the product of the matrices corresponding to x. Then we say that BP computes f if for all x, $B_x[1,w] = 0 \Leftrightarrow f(x) = 0$. In other words, the value of $f(x)$ is encoded by the top-right entry of B_x. We note that unlike prior works, the construction of BP in [AGIS14] does *not* use Barrington's theorem [Bar89], which is the source of their efficiency gains.

Once the BP is constructed, it is then randomized using Kilian's technique [Kil88]. Recall that for this we choose $n-1$ non-singular matrices $R_1, \dots, R_{n-1} \in \mathbb{Z}_p^{w \times w}$ uniformly at random, and set $\widetilde{B}_{i,b} := R_{i-1}^{-1} \cdot B_{i,b} \cdot R_i$ for each i and b. (For notational brevity we define $R_0 = R_n = I_{w \times w}$.) Next the obfuscator further randomizes the BP by choosing $2n$ non-zero scalars $\alpha_{i,b} \in \mathbb{Z}_p \setminus \{0\}$ uniformly and independently, and setting $C_{i,b} := \alpha_{i,b} \cdot \widetilde{B}_{i,b}$. It can be easily verified that the new BP $\{C_{i,b} \mid i, b\}$ computes the same function as before with probability 1. Building on [Kil88], [AGIS14] show that for every $x \in \{0,1\}^\ell$ the marginal distribution on $\{C_{i,x_{\mathsf{inp}(i)}} \mid i \in [n]\}$ can be efficiently computed given only x and $f(x)$.

The final step in the obfuscation is to encode the matrix elements using the graded encoding scheme. Recall that to do this, we must choose an index set $S \subseteq \mathbb{U}$ for each element. For this overview, the important points are that (1) within a single matrix all entries have the same index-set; (2) for any i, i' such that $\mathsf{inp}(i) = \mathsf{inp}(i')$, the index-sets for $C_{i,0}$ and $C_{i',1}$ have a non-empty intersection; and (3) for any $x \in \{0,1\}^\ell$, an element with index set $S = \mathbb{U}$ corresponding to the honest evaluation $C_x[1,w]$ can be efficiently computed using only the fully-restricted graded encoding scheme operations. Here we differ slightly from [AGIS14], in that their sets do not guarantee (2). For further details on the set system, see Section 2.2.

Simulating the graded encoding interface. To prove VBB security, we must show that for any poly-time adversary, the view resulting from its interaction with the graded encoding scheme can be efficiently simulated using only black-box access to the function f. Since the simulator does not have access to the branching program computing f, the first step is to simulate the initial elements by creating a unique formal variable for each entry of each matrix $C_{i,b}$ and assigning to each variable the corresponding index-set used by the obfuscator. Note that up to now the simulation is perfect, because in an ideal graded encoding scheme the adversary sees only random representations of the encoded elements. (This first step follows the strategy in [BR14b, BGK+14, AGIS14].)

In all prior works, simulating the arithmetic operations $+, -, \times$ was trivial, because in a fully-restricted graded encoding scheme two elements can be checked for compatibility by just looking at their index-sets. Here, simulating $+$ and $-$ is similarly trivial because these operations are always valid, but simulating $\times$ involves checking whether the product of two arithmetic circuits over the initial elements computes a valid polynomial. (Recall that a polynomial is valid iff the variables appearing in each non-zero monomial have pairwise disjoint index-sets.) To solve this we show that checking validity is reducible to identity-testing, and thus we can use the Schwartz-Zippel lemma to efficiently check validity up to a negligible error probability. This is done in two steps. First, we use a result of Fournier, Malod, and Mengel [FMM12] that testing for multilinearity reduces to identity testing. (Note that any non-multilinear polynomial is not valid.) Second, for each pair of variables $x \neq x'$ with intersecting index-sets, we construct a circuit that computes exactly those monomials in which both x and x' appear. If all such circuits are identically zero, the multiplication is valid.

We now turn to simulating the zero-test queries, which, as in prior works, makes up the bulk of the analysis. The simulator is given an arithmetic circuit e computing a valid polynomial over the initial elements, and needs to check if e evaluates to 0 on the obfuscated program.

Recall that the obfuscated matrices are computed as $C_{i,b} := \alpha_{i,b} \cdot \widetilde{B}_{i,b}$, where each $\alpha_{i,b}$ is uniform and independent in $\mathbb{Z}_p \setminus \{0\}$. Thus we may view e as a multilinear polynomial in the variables $\alpha_{i,b}$, where the coefficient of each "α-monomial" is a polynomial in the entries of the corresponding $\widetilde{B}_{i,b}$. This is useful because e evaluates to 0 on the obfuscation iff each of its α-monomials do. Furthermore the marginal distribution on each valid α-monomial can be simulated with only black-box access to f. This is because every valid α-monomial contains at most one $\alpha_{i,b}$ for each layer i (because of the index set construction mentioned above). Thus if it has degree $< n$ then the marginal distribution on its coefficient is just uniform non-singular matrices, and if it has degree $= n$ then it corresponds to a single input x and by construction the marginal distribution can be simulated given only $f(x)$. In summary, if we could efficiently decompose e into its non-zero α-monomials and show that there are at most $\mathsf{poly}(n)$ of them, then the zero-test could be efficiently simulated.

The approach of viewing e in this way was used in the prior works [BR14b, BGK+14, AGIS14].[2]

[2] The [BR14b] construction does not have exactly this form, but the analysis is the same in spirit.

Specifically, [BR14b] gives a procedure for decomposing e into its α-monomials which runs in time proportional to the number of such monomials, and then shows that under the Bounded Speedup Hypothesis the number of α-monomials is at most $\mathsf{poly}(n)$. [BGK+14] gives a different procedure (also used by [AGIS14]) for decomposing e into its degree-n α-monomials; they show that if e was constructed using a fully-restricted graded encoding scheme, then the decomposition runs in polynomial time and further that e only contains degree-n α-monomials.

In our setting, there are a number of obstacles. First, we wish to avoid the Bounded Speedup Hypothesis when possible. Second, there are elements e in a multiplication-restricted graded encoding scheme for which the [BGK+14] decomposition algorithm has a super-polynomial running time. Third, because we allow zero-testing at any level, we can no longer guarantee that e consists only of degree-n α-monomials.

We overcome these obstacles by giving a new decomposition algorithm, and we show that it runs in polynomial time on any e constructed in a multiplication-restricted graded encoding scheme. Our decomposition algorithm differs from previous works in several ways, one of which is that it does not isolate each α-monomial by itself, but rather returns a set of elements where each contains at most one degree-n α-monomial (and possibly other lower-degree monomials).

In each step, our decomposition takes a *global* view of the circuit, while previous algorithms took an arguably more local view. For example, proceeding downward through the circuit, when we reach a node whose leaves do not touch every layer of the branching program, we do not decompose it further because it cannot possibly contain a degree-n α-monomial. Similarly, when we reach a multiplication node whose leaves *do* touch every layer, we show that at most one of its children requires further decomposition, and we use arithmetic circuit tools to analyze the functions computed by the children to select the right one.

From the elements returned by the decomposition, the degree-n α-monomials can be fully extracted using the classical algorithm for computing the homogeneous degree-n portion of a circuit. We finally show that the set of all α-monomials with degree $< n$ can be *collectively* zero-tested using Schwartz-Zippel, because each is zero on the obfuscation iff it is the identically zero polynomial.

Unrestricted grading encoding schemes. We now discuss the changes that are needed for unrestricted graded encoding schemes. In this setting, it turns out that if every valid poly-size element e has a poly-size decomposition as above, then VP $\neq$ VNP (Theorem 5.3). Thus we cannot hope for an unconditional result.

We take the approach of Brakerski and Rothblum [BR14b] and bound the number of full α-monomials under a new parameterized assumption inspired by their Bounded Speedup Hypothesis. (The exact bound depends on the parameter in the assumption.) Once we have this bound on the number of full α-monomials, we apply essentially the algorithm from [BR14b] for zero-testing. One important difference is that here we cannot guarantee that e contains *only* full α-monomials (because we allow zero-testing at any level), but we adapt to this again by extracting the homogeneous degree-n portion of e to get just the full α-monomials.

Organization. In Section 2 we give some preliminaries.The analysis of the multiplication-restricted graded encoding scheme, and the proof of Theorem 1.6, appear in Section 4. In Section 5 we analyze the unrestricted graded encoding scheme and prove Theorem 1.7 and Corollary 1.8.

2 Preliminaries

2.1 Arithmetic circuit tools

We use several tools for analyzing and modifying arithmetic circuits, repeatedly using the classical algorithm for extracting the homogeneous degree-d portion of an arithmetic circuit. A proof can be found in, e.g., [Bür00, Lemma 2.14].

Lemma 2.1 (Extract homogeneous polynomial)**.** *There is an algorithm that, given an arithmetic circuit e of size $\mathsf{poly}(n)$ on n variables and an integer d, runs in time $\mathsf{poly}(n, d)$ and outputs a circuit of size $O(d^2 \cdot |e|)$ that computes the degree-d portion of e.*

Each of the following arithmetic circuit testing procedures is based on a reduction to identity-testing and an application of the Schwartz-Zippel lemma. We remark that these procedures test properties of the *formal expression* computed by an arithmetic circuit, and so we can apply the Schwartz-Zippel lemma over a sufficiently large field to get an algorithm with running time $\mathsf{poly}(n)$ and error probability $\mathsf{negl}(n)$.

Lemma 2.2 (Multilinearity check; [FMM12, Prop. 5.1])**.** *There is an algorithm that, given an arithmetic circuit e of size $\mathsf{poly}(n)$ on n variables, runs in time $\mathsf{poly}(n)$ and with probability $1 - \mathsf{negl}(n)$ correctly decides whether e computes a multilinear polynomial.*

Lemma 2.3 (Variable appearance check)**.** *There is an algorithm that, given an arithmetic circuit e of size $\mathsf{poly}(n)$ on n variables and a variable x of e, runs in time $\mathsf{poly}(n)$ and with probability $1 - \mathsf{negl}(n)$ correctly decides whether any non-zero monomial of e contains x.*

Proof. Let $e|_{x=0}$ be the circuit obtained from e by setting all instances of x to 0. Let $e^{(x)} := e - e|_{x=0}$ be the circuit computing exactly the set of non-zero monomials from e in which x appears. Then $e^{(x)} \equiv 0$ iff x appears in no non-zero monomial of e. $\square$

Lemma 2.4 (Variable multiplication check)**.** *There is an algorithm that, given an arithmetic circuit e of size $\mathsf{poly}(n)$ on n variables and two variables $x \neq x'$ of e, runs in time $\mathsf{poly}(n)$ and with probability $1 - \mathsf{negl}(n)$ correctly decides whether any non-zero monomial of e contains both x and x'.*

Proof. Let $e' := e^{(x)}$ where $e^{(x)}$ is as in the previous lemma. Then $e'^{(x')} \equiv 0$ iff no non-zero monomial of e contains both x and x'. $\square$

2.2 Strong straddling sets

In this section we define the notion of *strong* straddling set systems, which strengthen the straddling set systems introduced by Barak et al. [BGK+14] by adding an additional "strong intersection" property.

Definition 2.5 (Strong straddling set system)**.** A *strong straddling set system* with n entries is a collection of sets $\mathbb{S}_n = \{S_{i,b} : i \in [n], b \in \{0,1\}\}$ over a universe $\mathbb{U}$, such that $\cup_{i\in[n]} S_{i,0} = \mathbb{U} = \cup_{i\in[n]} S_{i,1}$, and the following holds.

- (Collision at universe.) If $C, D \subseteq \mathbb{S}_n$ are distinct non-empty collections of disjoint sets such that $\bigcup_{S\in C} S = \bigcup_{S\in D} S$, then $\exists b \in \{0,1\}$ such that $C = \{S_{i,b}\}_{i\in[n]}$ and $D = \{S_{i,1-b}\}_{i\in[n]}$.
- (Strong intersection.) For every $i, j \in [n]$, $S_{i,0} \cap S_{j,1} \neq \emptyset$.

We refer to sets of the form $S_{i,0}$ ($S_{i,1}$) as "0-sets" ("1-sets"). We can construct a strong straddling set system for every n, as follows.

Construction 2.6 (Strong straddling set system)**.** Let $\mathbb{S}_n = \{S_{i,b} : i \in [n], b \in \{0,1\}\}$ over a universe $\mathbb{U} = \{1, 2, ..., n^2\}$, where

$$S_{i,0} = \{n(i-1)+1, n(i-1)+2, \ldots, ni\} \text{ and } S_{i,1} = \{i, n+i, 2n+i, \ldots, n(n-1)+i\}$$

for all $1 \le i \le n$.

The strong intersection property is obtained by taking a "two-dimensional" view of the universe $\mathbb{U} = \{1, ..., n^2\}$, whereas previous straddling set systems [BBC+14, AGIS14] were in some sense "one-dimensional".

2.3 The ideal graded encoding model

In this section we describe the ideal graded encoding model which is used by the obfuscator and evaluator. This model is exactly analogous to the ideal graded encoding model of [BGK+14], but with their fully-restricted graded encoding scheme replaced by our two new graded encoding schemes (Definitions 1.3 and 1.4).

In the ideal graded encoding model, we have an oracle $\mathcal{M}$ that implements an idealized version of a graded encoding scheme. $\mathcal{M}$ maintains a list of elements, and allows a user to perform valid arithmetic operations over these elements. $\mathcal{M}$ maintains a table that maps elements to generic representations called *handles*. Each handle is generated uniformly at random subject to being distinct for all other handles. (Note that each handle is distinct even if the same element appears multiple times in the table.) The user sees only the handles, and may query $\mathcal{M}$ with them to evaluate the operations of the graded encoding scheme ($+$, $-$, $\times$, and IsZero).

$\mathcal{M}$ is initialized with a set of basic elements $\{(r_i, S_i)\}_i$, and generates a handle for each basic element. Then given two handles h_1, h_2 and an operation $\circ \in \{+, -, \times\}$, $\mathcal{M}$ first looks up the corresponding elements e_1, e_2 in the table. If either does not exist, or if $e_1 \circ e_2$ is not permitted by the graded encoding scheme, the call fails. Otherwise $\mathcal{M}$ generates a new handle for $e_1 \circ e_2$, saves this in the table, and returns the new handle. Calls to IsZero are evaluated analogously, but for these $\mathcal{M}$ returns 0 or 1 instead of a new handle.

2.4 Relaxed matrix branching programs

Our obfuscator will first transform the input formula F into a *dual-input, oblivious, relaxed matrix branching program (RMBP)*, which will then be obfuscated. RMBPs were introduced in [AGIS14], who use dual-input oblivious RMBPs towards improving the efficiency of candidate obfuscators.

Definition 2.7 (Dual-input RMBP)**.** Let $\mathcal{R}$ be any finite ring. A *dual-input relaxed matrix branching program* (over $\mathcal{R}$) of size w and length n for ℓ-bit inputs is given by a sequence $\mathsf{BP} = (\mathsf{inp}_1, \mathsf{inp}_2, B_{i,b_1,b_2})_{i\in[n],b_1,b_2\in\{0,1\}}$, where each B_{i,b_1,b_2} is a $w \times w$ full-rank matrix, and $\mathsf{inp}_1, \mathsf{inp}_2 : [n] \to [\ell]$ are the evaluation functions of BP. The output of BP on input $x \in \{0,1\}^\ell$, denoted by $\mathsf{BP}(x)$, is determined as follows $\mathsf{BP}(x) = 1$ if and only if $\left(\prod_{i=1}^{n} B_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right)[1,w] \neq 0$. We say that a set of dual-input RMBPs is *oblivious* if the functions $\mathsf{inp}_1, \mathsf{inp}_2$ depend only on n and ℓ (and not on the function being computed).

2.5 Obfuscation security in an idealized model

We now define Virtual Black-Box (VBB) Obfuscation, and Indistinguishability Obfuscation (iO) in the $\mathcal{M}$-idealized model, where $\mathcal{M}$ is some oracle. In this model, both the obfuscator and the evaluator have access to the oracle $\mathcal{M}$. However, the function family that is being obfuscated does not have access to $\mathcal{M}$.

Definition 2.8. For a (possibly randomized) oracle $\mathcal{M}$, and a circuit class $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$, we say that a uniform PPT oracle machine $\mathcal{O}$ is a "Virtual Black-Box" Obfuscator for $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ in the $\mathcal{M}$-idealized model, if the following conditions are satisfied:

- Functionality: For every $\ell \in \mathbb{N}$, every $C \in \mathcal{C}_\ell$, every input x to C, and for every possible coins for $\mathcal{M}$:

$$\Pr[(\mathcal{O}^{\mathcal{M}}(C))(x) \neq C(x)] \leq \mathrm{negl}(|C|) \ ,$$

  where the probability is over the coins of $\mathcal{O}$.

- Polynomial Slowdown: There exist a polynomial p such that for every $\ell \in \mathbb{N}$ and every $C \in \mathcal{C}_\ell$, we have that $|\mathcal{O}^{\mathcal{M}}(C)| \leq p(|C|)$.

- Virtual Black-Box: For every PPT adversary $\mathcal{A}$ there exists a PPT simulator Sim such that for all PPT distinguishers D, all $\ell \in \mathbb{N}$, and all $C \in \mathcal{C}_\ell$:

$$\left| \Pr[D(\mathcal{A}^{\mathcal{M}}(\mathcal{O}^{\mathcal{M}}(C))) = 1] - \Pr[D(\mathsf{Sim}^{C}(1^{|C|})) = 1] \right| \leq \mathsf{negl}(|C|)$$

  where the probabilities are over the coins of D, $\mathcal{A}$, Sim, $\mathcal{O}$, and $\mathcal{M}$.

Indistinguishability obfuscation is similar to VBB obfuscation, except that the simulator is not required to be efficient.

Definition 2.9. For $\mathcal{M}$ and $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ as in Definition 2.8, we say that $\mathcal{O}$ is an Indistinguishability Obfuscator (iO) for $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ in the $\mathcal{M}$-idealized model, if it satisfies the functionality and polynomial slowdown properties as in Definition 2.8, and an *indistinguishability* property which is identical to the VBB property except that there is no bound on the running time of Sim.

3 Obfuscation in the ideal graded encoding model

We describe an obfuscator $\mathcal{O}$ for NC^1 circuits in the ideal graded encoding model. The obfuscator is identical to the obfuscator of [AGIS14], except that it encodes elements using strong (as opposed to standard) straddling set systems.

On input an NC^1 circuit $F : \{0,1\}^\ell \to \{0,1\}$, the obfuscator $\mathcal{O}$ first converts F into an oblivious dual-input RMBP as described in [AGIS14, Section 3]. This RMBP is denoted $\mathsf{BP} = \left(\mathsf{inp}_1, \mathsf{inp}_2, \{B_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}\right)$, where $\mathsf{inp}_1, \mathsf{inp}_2 : [n] \to [\ell]$ are evaluation functions, each $B_{i,b_1,b_2} \in \{0,1\}^{w\times w}$ has full rank, and the following holds.

1. $\mathsf{inp}_1(i) \neq \mathsf{inp}_2(i)$ for every $i \in [n]$.
2. For every $(j,k) \in [\ell] \times [\ell]$, there exists an index $i \in [n]$ such that $\mathsf{inp}_1(i) = j \wedge \mathsf{inp}_2(i) = k$, or $\mathsf{inp}_1(i) = k \wedge \mathsf{inp}_2(i) = j$. (That is, every pair of input bits are paired at some layer of the BP.)
3. For every $j \in [\ell]$, let $\mathsf{ind}(j) := \{i \in [n] : \mathsf{inp}_1(i) = j\} \cup \{i \in [n] : \mathsf{inp}_2(i) = j\}$. Then there exists an $\ell' \in \mathbb{N}$ such that $|\mathsf{ind}(j)| = \ell'$ for every $j \in \ell'$.

Randomizing BP. $\mathcal{O}$ samples a large enough prime p with $\Omega(n)$ bits, and randomizes BP following [AGIS14, Section 4]. Specifically, $\mathcal{O}$ generates $\left(\tilde{s}, \{C_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}, \tilde{t}\right) := \mathsf{randBP}(\mathsf{BP})$ as follows.

1. Choose $n+1$ uniform and independent full-rank matrices $R_0, ..., R_n \in \mathbb{Z}_p^{w\times w}$, and set $\tilde{B}_{i,b_1,b_2} := R_{i-1} \cdot B_{i,b_1,b_2} \cdot R_i^{-1}$, for every $i \in [n]$ and $b_1, b_2 \in \{0,1\}$.
2. Choose $4n$ uniform and independent non-zero scalars $\alpha_{i,b_1,b_2} \in \mathbb{Z}_p \setminus \{0\}$, and set $C_{i,b_1,b_2} := \alpha_{i,b_1,b_2} \cdot \tilde{B}_{i,b_1,b_2}$ for every $i \in [n]$ and $b_1, b_2 \in \{0,1\}$.
3. Set $\tilde{s} := e_1 \cdot R_0^{-1}$ and $\tilde{t} := R_n \cdot e_w$.

The obfuscation of F will consist of ideal encodings of the entries of $\tilde{s}, \tilde{t}$ and $\{C_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}$, with respect to the following strong straddling set systems.

Encoding the randomized BP. Let $\mathbb{U}$ be a universe set, and let $\mathbb{U}_s, \mathbb{U}_t, \mathbb{U}_1, ..., \mathbb{U}_\ell$ be a partition of $\mathbb{U}$ such that $|\mathbb{U}_j| = 2\ell' - 1$ for every $j \in [\ell]$. For $j \in [\ell]$, let $\mathbb{S}^j$ be a strong straddling set system with ℓ' entries over universe $\mathbb{U}_j$. We associate the sets in $\mathbb{S}^j$ with the layers i of the BP that are indexed by x_j (i.e., layers i such that $j \in \{\mathsf{inp}_1(i), \mathsf{inp}_2(i)\}$) as follows: $\mathbb{S}^j = \left\{S^j_{k,b} :\ k \in \mathsf{ind}(j), b \in \{0,1\}\right\}$. Next, we associate an index-set with every entry of $\{C_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}$, as follows. The set $S(i,b_1,b_2) := S^{\mathsf{inp}_1(i)}_{i,b_1} \cup S^{\mathsf{inp}_2(i)}_{i,b_2}$ is associated with C_{i,b_1,b_2}, and will be used to encode the entries $C_{i,b_1,b_2}[k,l]$ of C_{i,b_1,b_2} (where $C_{i,b_1,b_2}[k,l]$ denotes the (k,l)'th entry of C_{i,b_1,b_2}). $\mathbb{U}_s, \mathbb{U}_t$ will encode the entries of $\tilde{s}, \tilde{t}$, respectively. More formally, $\mathcal{O}$ initializes the oracle $\mathcal{M}$ with the ring $\mathbb{Z}_p$, the universe set $\mathbb{U}$, and the following set X of variables

$$\left\{\{(\tilde{s}_i, \mathbb{U}_s)\}_{i\in[w]}, \{(\tilde{t}_i, \mathbb{U}_t)\}_{i\in[w]}, \{(C_{i,b_1,b_2}[k,l], S(i,b_1,b_2))\}_{i\in[n],b_1,b_2\in\{0,1\},k,l\in[w]}\right\}$$

where by (x, S) we mean that the index-set associated with x is $\{S\}$, for $S \subseteq \mathbb{U}$. The oracle $\mathcal{M}$ returns handles to these elements, and $\mathcal{O}$ outputs these handles as the obfuscation of F.

Remark 3.1. Notice that the obfuscation includes the vectors $\tilde{s}, \tilde{t}$, which were omitted from the informal description of $\mathcal{O}$ in Section 1.3. We note that the simulator (of Sections 4 and 5) generates handles also for the entries of $\tilde{s}, \tilde{t}$, with the corresponding index-sets (just as it does for the entries of the C_{i,b_1,b_2}'s), and that in the description of an element e as a polynomial in the α_{i,b_1,b_2}'s, the coefficients are polynomial in the entries of the $\tilde{B}_{i,b_1,b_2}$'s, *as well as the entries of* $\tilde{s}, \tilde{t}$. This, however, does not influence the analysis.

4 VBB security for multiplication-restricted graded encodings

In this section, we show that given an obfuscation of an NC^1 function F, there exists an efficient simulator that can answer the queries of any polynomially-bounded adversary in an ideal multiplication-restricted graded encoding scheme, such that the simulated answers are statistically close to the answers given by the ideal oracle $\mathcal{M}$.

The simulator Sim is given $1^{|F|}$ and a description of the adversary $\mathcal{A}$, and has oracle access to F. To simulate the obfuscator $\mathcal{O}$, Sim generates formal variables representing each entry of the matrices $\{C_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}$ (including their index sets), and generates handles corresponding to these elements. Sim maintains a table of handles, and simulates $\mathcal{A}$'s oracle calls to $\mathcal{M}$. Addition and subtraction queries can be simulated trivially since there are no constraints on these operations. Next we describe how Sim simulates multiplication queries and zero-test queries.

4.1 Simulating Multiplication Queries

To answer a multiplication query the simulator must check, given two arithmetic circuits e_1 and e_2, whether the circuit $e := e_1 \times e_2$ computes a valid polynomial. Recall (Def. 1.2) that a polynomial is valid if the basic elements appearing in any monomial have pairwise disjoint index sets.

Let X be the set of all basic elements that appear in either e_1 or e_2. Then the validity check has two steps. First we verify that e is multilinear, i.e. that no monomial in e contains multiple copies of some $x \in X$. This is done using the algorithm from Lemma 2.2. Second we verify that for each $x \neq x' \in X$ with intersecting index-sets, no monomial of e contains both x and x'. This is done using the algorithm from Lemma 2.4.

If the query is valid, then Sim generates a new handle h for e, adds it to the handle-set, and returns h to the adversary as the answer to the query. The proof of the next lemma is immediate given Lemmas 2.2 and 2.4 (which say that both the multilinearity check and the variable multiplication check run in time $\mathsf{poly}(n)$ and return the correct answer with probability $1 - \mathsf{negl}(n)$).

Lemma 4.1. *For every multiplication query $e_1 \times e_2$ of a polynomially-bounded adversary $\mathcal{A}$, Sim runs in polynomial time and generates an answer that is $(1 - \mathsf{negl}(n))$-close to the real-world answer.*

4.2 Simulating Zero-Test Queries

In this section we describe how the simulator Sim answer a single zero-test on an element e. We use the following terminology, adapted from [BGK+14, AGIS14].

Definition 4.2 (Touching matrices and layers)**.** We say that an element e *touches a matrix* $C_{i,b_1,b_2}, i \in [n], b_1, b_2 \in \{0,1\}$ if some non-zero monomial in e contains a variable representing an entry of C_{i,b_1,b_2}. We say that e *touches layer* i if it touches a matrix C_{i,b_1,b_2} for some $b_1, b_2 \in \{0,1\}$.

Next, we define the notion of an input-profile of an element e, which represents the partial information that e gives about the input $x \in \{0,1\}^\ell$. We also define single-input elements as elements whose arithmetic circuit depends on formal variables that all correspond to a *single* input x.

Definition 4.3 (Input-profiles and single-input elements)**.** For an element e, its *input-profile* $\mathsf{Prof}(e) \in \{0,1,*\}^\ell \cup \{\perp\}$ is defined as follows. For $j \in [\ell]$, we say that $\mathsf{Prof}(e)_j$ is *consistent with* $b \in \{0,1\}$ if e touches any matrix C_{i,b_1,b_2} such that $\mathsf{inp}_l(i) = j$ and $b_l = b$ for some $l \in \{1,2\}$. If $\mathsf{Prof}(e)_j$ is consistent with b, but not with $1-b$, then we set $\mathsf{Prof}(e)_j := b$. If $\mathsf{Prof}(e)_j$ is not consistent with either of $b, 1-b$ then we set $\mathsf{Prof}(e)_j = *$. If $\mathsf{Prof}(e)_j$ is consistent with both $b, 1-b$, then we say that e *conflicts on index* j. If e conflicts on some index $j \in [\ell]$, then we set $\mathsf{Prof}(e) = \perp$, and call $\mathsf{Prof}(e)$ *invalid.* A *single-input element* is an element e with $\mathsf{Prof}(e) \neq \perp$. We say that e has a *complete* profile if $\mathsf{Prof}(e) \in \{0,1\}^\ell$, otherwise $\mathsf{Prof}(e)$ is *incomplete.* A pair e, e' of elements *conflict on index* j if $\mathsf{Prof}(e)_j = 1 - \mathsf{Prof}(e')_j$ or either $\mathsf{Prof(e)}$ or $\mathsf{Prof(e')}$ is invalid.

Note that $\mathsf{Prof(e)}$ can be computed (up to negligible error probability) in time $\mathsf{poly}(|e|)$, by using the algorithm from Lemma 2.3 that checks which variables appear in e's non-zero monomials.

At a high level, the simulator answers a zero-test query "$e = 0$?" as follows. First, it decomposes e into a list $\{e_1, ..., e_m\}$ of elements, such that: $e = \sum_{i=1}^m e_i$, in the sense that both sides of the equation compute the same function (but, possibly, using different arithmetic circuits); every e_i is a single-input element, or does not touch all layers $i \in [n]$; and $m = \mathsf{poly}(n)$. Then, the simulator extracts the full α-monomials from the single-input elements, and performs a zero-test on each separately. Finally, it performs a zero-test on the remains of e (i.e. on the non-full α-monomials). Next, we describe the decomposition algorithm.

4.2.1 Decomposition Algorithm

We describe a decomposition $D(e)$ of an element e, satisfying the three properties described in Figure 1.

1. $e = \sum_{s \in D(e)} s$, where equivalence is as a polynomial function.
2. $\forall s \in D(e)$: s is either single-input or does not touch every layer.
3. $|D(e)| \leq \mathsf{poly}(|e|)$.

Figure 1: Properties of a valid decomposition $D(e)$

Theorem 4.4. *For any valid element e, there exists a $\mathsf{poly}(|e|)$-time computable decomposition $D(e)$ satisfying the properties in Figure 1.*

Recall that e is given as a fan-in-2 arithmetic circuit. For the decomposition, we instead view e as a layered unbounded fan-in circuit whose layers alternate between addition (or subtraction) and multiplication gates, and we assume that all input wires to a layer come from the layer directly below. Any e can be converted to such a circuit with at most a $\mathsf{poly}(|e|)$ increase in size. For the remainder, we refer to the layers of e as *sections* to avoid confusion with layers of the branching program. We assume without loss of generality that the top section of e contains multiplication gates.

We compute the decomposition by starting with $D(e) = \{e\}$, and then refining until the properties in Figure 1 are satisfied. We keep two lists GO and STOP, where each list contains pairs of arithmetic expressions (z, z'). GO contains expressions that need to be further refined, and STOP contains expressions that do not. We terminate when GO $= \emptyset$ and then set $D(e) := \{zz' \mid (z, z') \in \text{STOP}\}$.

Throughout the decomposition, we maintain the invariants shown in Figure 2. Letting m denote the number of sections in e, we label the sections, starting from 1 at the top, so the m^{th} section contains the basic elements at e's input.

We remark that the set of layers touched by any expression e can be computed in time $\mathsf{poly}(|e|)$ using the algorithm from Lemma 2.3.

Lemma 4.5. *For any algorithm satisfying the invariants in Figure 2, upon termination the decomposition $D(e) = \{zz' \mid (z, z') \in STOP\}$ satisfies the properties in Figure 1.*

Proof. We show that GO $= \emptyset$ after m steps. Then invariants 1, 2, and 4 imply the three decomposition properties, respectively.

Any gate in section m is a single-input element, and a valid product of two single-input elements is also single-input by Lemma 4.7. Thus, after step m GO cannot contain any (z, z') that satisfies the third invariant, so GO $= \emptyset$. □

Next, we prove Theorem 4.4.

Proof of Theorem 4.4. We give an m-step $\mathsf{poly}(|e|)$-time algorithm satisfying the invariants in Figure 2. In step 1, if e is single-input or does not touch every layer, then we set GO $= \emptyset$ and STOP $= \{(1, e)\}$. Otherwise we set GO $= \{(1, e)\}$ and STOP $= \emptyset$.

1. $e = \sum_{(z,z')\in \mathrm{GO}} zz' + \sum_{(z,z')\in \mathrm{STOP}} zz'$.
2. For each $(z, z') \in \mathrm{STOP}$, zz' is a valid element that is either single-input or does not touch every layer.
3. After step i, for each $(z, z') \in \mathrm{GO}$: z is single-input, z' is a gate in section i, and zz' is a valid element that touches every layer and is not single-input. Further, GO contains at most one (z, z') for each gate z' in section i.
4. During step i, $|\mathrm{STOP}|$ increases by at most the number of wires leaving section i.

Figure 2: Invariants of the decomposition algorithm

In step i ($2 \le i \le m$), we proceed as follows.

If section $(i-1)$ contains multiplication gates, then at the start of step i each $(z, z') \in \mathrm{GO}$ is of the form $(z, q_1 \times \ldots \times q_k)$ for some gates $q_1, \ldots, q_k$ in section i. Lemma 4.8 shows that there is a unique j^* such that q_{j^*} is not single-input, and we can find this j^* in time $\mathsf{poly}(|e|)$ by computing each $\mathsf{Prof}(q_j)$. So, we replace each $(z, q_1 \times \ldots \times q_k) \in \mathrm{GO}$ with $(z \times \prod_{j \neq j^*} q_j,\ q_{j^*})$. By Lemma 4.7, we have that $z \times \prod_{j \neq j^*} q_j$ is single-input. Further q_{j^*} is a gate in section i, and $z \times \prod_{j \le k} q_j$ touches every layer and is not single-input by the invariants on step $(i-1)$. Finally, to ensure that there is at most one $(z, z') \in \mathrm{GO}$ for each gate z' in section i, we repeatedly replace any $(z_1, z'), (z_2, z') \in \mathrm{GO}$ with $(z_1 + z_2, z')$. Lemma 4.9 shows that any such $z_1 + z_2$ is a single-input element, so the invariants remain satisfied.

If section $(i-1)$ contains addition gates, then at the start of step i each $(z, z') \in \mathrm{GO}$ is of the form $(z, q_1 + \cdots + q_k)$ for some gates $q_1, \ldots, q_k$ in section i. We first modify the expression $(q_1 + \cdots + q_k)$ by zeroing any basic elements that $(q_1 + \cdots + q_k)$ does not touch (in the sense of Definition 4.2), thus ensuring that zq_j is a valid element for each $j \le k$. Then for each such $(z, q_1 + \cdots + q_k)$, we remove it from GO and set

$$\mathrm{GO} \leftarrow \mathrm{GO}\ \cup\ \{(z, q_j) \mid zq_j \text{ touches every layer and is not single-input}\}$$

$$\mathrm{STOP} \leftarrow \mathrm{STOP}\ \cup\ \{(z, q_j) \mid zq_j \text{ does not touch every layer or is single-input}\}.$$

This adds at most one pair to STOP for each wire between layers i and $i-1$. Thus all invariants are now satisfied except that GO may contain multiple (z, z') for each gate z' in section i; to fix this, we again replace any $(z_1, z'), (z_2, z') \in \mathrm{GO}$ with $(z_1 + z_2, z')$. $\square$

We now prove the lemmas that were used in the proof of Theorem 4.4. Their proofs heavily rely on the strong intersection property of the strong straddling set system, and the structure of the dual-input RMBP, in which every pair of input bits are read together at some layer. Given an element e, we use $\mathcal{V}(e)$ to denote the set of variables that appear in e's non-zero monomials. We need the following structural result on multilinear polynomials.

Lemma 4.6. *Let e_1 and e_2 be arithmetic circuits computing multilinear polynomials. If $e := e_1 \times e_2$ is multilinear, then for all $x \in \mathcal{V}(e_1)$ and $y \in \mathcal{V}(e_2)$, e has a monomial that contains both x and y.*

Proof. We first show that if e is multilinear then $\mathcal{V}(e_1) \cap \mathcal{V}(e_2) = \emptyset$. If not, there is some $x \in \mathcal{V}(e_1) \cap \mathcal{V}(e_2)$. Then write

$$e_1 = x \cdot e_1' + e_1'' \qquad\qquad e_2 = x \cdot e_2' + e_2''$$

where e_1', e_1'', e_2', e_2'' all do not contain x and $e_1', e_2' \neq 0$. Then because the $x^2 \cdot e_1' \cdot e_2'$ term of e is non-zero and not cancelled by any other term, e is not multilinear.

We now have that $\mathcal{V}(e_1) \cap \mathcal{V}(e_2) = \emptyset$. For any $x \in \mathcal{V}(e_1), y \in \mathcal{V}(e_2)$, write

$$e_1 = x \cdot e_1' + e_1'' \qquad\qquad e_2 = y \cdot e_2' + e_2''$$

where e_1', e_1'', e_2', e_2'' all contain neither x nor y and $e_1', e_2' \neq 0$. Then similarly e must contain a monomial with xy. □

Lemma 4.7. *If e_1,e_2 are valid single-input elements and $e_1 \times e_2$ is valid, then $e_1 \times e_2$ is single-input.*

Proof. If $e_1 \times e_2$ conflicts on some index j, then there are basic sub-elements $x_1 \in \mathcal{V}(e_1)$ and $x_2 \in \mathcal{V}(e_2)$ that conflict on index j, and cannot be multiplied. But by Lemma 4.6, x_1 and x_2 appear together in some monomial, so $e_1 \times e_2$ is not valid. □

Lemma 4.8. *Let $e_1, \ldots, e_d$ be valid elements such that $\prod_{i \leq d} e_i$ is valid, touches every layer, and is not single-input. Then there is a unique i such that e_i is not single-input.*

Proof. First note that $\{\mathcal{V}(e_i) \mid i \in [d]\}$ gives a partition of all layers, identifying a variable with the layer in which it appears. This is by Lemma 4.6, because any two variables from the same layer cannot be multiplied.

Pick any i such that e_i is not single-input (there must be one by Lemma 4.7). Fix some index j such that e_i conflicts on j. Then e_i must touch every layer that reads index j. If not, then some other $e_{i'}$ touches a matrix C_{l,b_1,b_2} such that $\mathsf{inp}_k(l) = j$ (for some $k \in \{1, 2\}$), and (without loss of generality) $b_k = 0$, but then e_i and $e_{i'}$ could not be multiplied, because they conflict on index j.

If there is another value $i' \neq i$ such that $e_{i'}$ conflicts on index $j' \neq j$, then by the same argument $e_{i'}$ touches every layer that reads bit j'. But then any layer reading both j and j' is touched by both e_i and $e_{i'}$, which is a contradiction. □

Lemma 4.9. *Let $z_1 \times z'$ and $z_2 \times z'$ be valid elements that touch every layer and are not single-input. If z_1 and z_2 are each single-input, then so is $z_1 + z_2$.*

Proof. Assume for contradiction that z_1 and z_2 are single-input but $z_1 + z_2$ is not. Fix some j such that $z_1 + z_2$ conflicts on index j. Then without loss of generality we have that $\mathsf{Prof}(z_1)_j = 0$ and $\mathsf{Prof}(z_2)_j = 1$. As in the proof of Lemma 4.8, because z_1 is single-input we must have that z' touches every layer that reads an index on which $z_1 \times z'$ has a conflict. Since $z_1 \times z'$ has a conflict on at least one index, and since each pair of indices are read together in at least one layer, z' must touch some layer that reads index j. But then at least one of z_1 or z_2 must conflict with z', so either $z_1 \times z'$ or $z_2 \times z'$ is invalid. □

4.2.2 The Zero-Test Simulator

In this section we describe and analyze the simulator Sim^0 that is used to answer a single zero-test. Recall that an element e is an arithmetic circuit computing a polynomial whose variables are the entries of C_{i,b_1,b_2} for $i \in [n], b_1, b_2 \in \{0, 1\}$. However, as $C_{i,b_1,b_2} = \alpha_{i,b_1,b_2} \cdot \tilde{B}_{i,b_1,b_2}$, we can think of it as a polynomial in the α_{i,b_1,b_2}, with coefficients that are polynomials in the entries of $\tilde{B}_{i,b_1,b_2}$. Under this viewpoint, we refer to the monomials as "α-monomials". We associate an index-set with each α_{i,b_1,b_2} and each entry of $\tilde{B}_{i,b_1,b_2}$, namely the index-set of C_{i,b_1,b_2}.

Definition 4.10. We say that a monomial in the variables $\{\alpha_{i,b_1,b_2} : i \in [n], b_1, b_2 \in \{0,1\}\}$ is *full* if it contains, for every $i \in [n]$, exactly one of the α's of layer i (i.e., one of $\alpha_{i,0,0}, \alpha_{i,0,1}, \alpha_{i,1,0}, \alpha_{i,1,1}$).

Notice that if e is valid then every α-monomial contains at most one α from every layer, because the index-sets of every pair of layer-i α's intersect and so they cannot be multiplied. We need the following simple observation.

Lemma 4.11. *Let e be a valid element and let $D(e)$ be its decomposition given by Theorem 4.4. Then each $s \in D(e)$ contains at most one full α-monomial, and each of e's full α-monomials appears in exactly one $s \in D(e)$.*

Proof. We may assume without loss of generality that each single-input element in $D(e)$ has a unique profile, by replacing any $s \neq s' \in D(e)$ such that $\mathsf{Prof}(s) = \mathsf{Prof}(s') \neq \perp$ with $s + s'$. Then the lemma holds because (1) any element that does not touch every layer cannot contain a full α-monomial, and (2) any single-input element s with a complete profile can only contain the unique full α-monomial corresponding to $\mathsf{Prof}(s)$. (Note that (1) includes single-input elements with incomplete profiles.) ☐

Given an element s that contains at most one full α-monomial, we can extract it (if it exists) by computing the homogeneous degree-n portion of s using the algorithm from Lemma 2.1. This is because in a valid polynomial, the only monomials of degree n are the full α-monomials. Further, we show in Lemma 4.16 below that for any element s with *no* full α-monomials, with high probability s evaluates to 0 on the obfuscation iff it computes the identically 0 polynomial.

The final ingredient we need is a method of sampling an assignment to the variables of a full α-monomial that is indistinguishable from the corresponding marginal distribution of the obfuscation. We use the method of [AGIS14, Thm. 7] (recall that randBP was defined in Section 3).

Theorem 4.12 ([AGIS14])**.** *Let* BP *be an oblivious dual-input RMBP that computes* $F : \{0,1\}^\ell \to \{0,1\}$*, and let* $\mathsf{BP}' := \mathsf{randBP}(\mathsf{BP})$*. There exists a PPT simulator* Sim' *such that for every* $x \in \{0,1\}^\ell$*,* $\left\{\mathsf{BP}'|_x\right\} \equiv \left\{\mathsf{Sim}'\left(1^{|F|}, F(x)\right)\right\}$*.*

We are now ready to describe the simulator.

Construction 4.13 (Zero-test simulator)**.** The zero-test simulator Sim^0 uses the decomposition algorithm D of Theorem 4.4. On input a valid element e, Sim^0 operates as follows.

1. Compute the decomposition $D(e)$.
2. For every single-input element $s \in D(e)$ with a complete profile, use Lemma 2.1 to construct an element $\widetilde{\alpha}_s$ that computes the homogeneous degree-n portion of s. (If s is not single-input or has an incomplete profile, define $\widetilde{\alpha}_s := 0$.)
3. For every single-input element $s \in D(e)$ with a complete profile, zero-test $\widetilde{\alpha}_s$ as follows: query the oracle F on $x := \mathsf{Prof}(\mathsf{s})$, and evaluate $\widetilde{\alpha}_s$ on $\mathsf{Sim}'\left(1^{|F|}, F(x)\right)$, where Sim' is the simulator of [AGIS14, Thm. 7]. If any such evaluation is non-zero, stop and return "$e \neq 0$".
4. Construct the element $e' := e - \sum_{s \in D(e)} \widetilde{\alpha}_s$, and test if e' computes the identically zero polynomial using Schwartz-Zippel. If so then return "$e = 0$", otherwise return "$e \neq 0$".

Construction 4.13 runs in time $\mathsf{poly}(n)$ because each step does. The following theorem shows its correctness, and completes the proof of Theorem 1.6. We use $\mathcal{V}^{\mathsf{real}}$ to denote the real-world distribution of the obfuscated program, and $e(\mathcal{V}^{\mathsf{real}}) \equiv 0$ to denote that e evaluates to 0 on the support of $\mathcal{V}^{\mathsf{real}}$.

Theorem 4.14. *Let e be a valid element, and let Sim^0 be as in Construction 4.13. Then*

$$\left|\Pr\left[\mathsf{Sim}^0\left(e\right)=0\right]-\Pr_{v\leftarrow\mathcal{V}^{\mathsf{real}}}\left[e\left(v\right)=0\right]\right|=\mathsf{negl}(n)$$

where the probabilities are over the randomness of Sim^0 and the obfuscator.

Proof. Lemma 4.15 below shows that for any valid element e, if $e(\mathcal{V}^{\mathsf{real}})\not\equiv 0$ then $\Pr_{v\leftarrow\mathcal{V}^{\mathsf{real}}}\left[e\left(v\right)=0\right]=\mathsf{negl}(n)$. (This is proved exactly as in [BGK+14].) Thus it suffices to prove that, with high probability over its randomness, Sim^0 returns "$e=0$" iff $e(\mathcal{V}^{\mathsf{real}})\equiv 0$. Observe further that $e(\mathcal{V}^{\mathsf{real}})\equiv 0$ if and only if $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}})\equiv 0$ for every α-monomial $\widetilde{\alpha}$ in e. The "if" direction is clear; for the "only if" direction, assume that some α-monomials are not identically zero on $\mathcal{V}^{\mathsf{real}}$. Then for some sample of the marginal distribution on $\{\widetilde{B}_{i,b_1,b_2}\}_{i,b_1,b_2}$, e becomes a non-zero polynomial in just the variables $\{\alpha_{i,b_1,b_2}\}_{i,b_1,b_2}$. Then since the marginal distribution on this latter set is uniform conditioned on any sample of $\{\widetilde{B}_{i,b_1,b_2}\}_{i,b_1,b_2}$, there is some sample $v\leftarrow\mathcal{V}^{\mathsf{real}}$ for which $e(v)\neq 0$, and thus $e(\mathcal{V}^{\mathsf{real}})\not\equiv 0$.

We now show that, with probability $1-\mathsf{negl}(n)$ over its randomness, Sim^0 returns "$e=0$" iff all α-monomials $\widetilde{\alpha}$ in e satisfy $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}})\equiv 0$.

Assume that e contains some full α-monomial $\widetilde{\alpha}_s$ such that $\widetilde{\alpha}_s(\mathcal{V}^{\mathsf{real}})\not\equiv 0$. We claim that, with probability $1-\mathsf{negl}(n)$ over the randomness of Sim^0, step 3 in Construction 4.13 returns "$e\neq 0$". Indeed, because the call to Sim' generates exactly the marginal distribution on $\widetilde{\alpha}_s$'s variables by Theorem 4.12, the evaluation generates a sample from $\widetilde{\alpha}_s(\mathcal{V}^{\mathsf{real}})$. By Lemma 4.15 this evaluation is non-zero with probability $1-\mathsf{negl}(n)$ because $\widetilde{\alpha}_s(\mathcal{V}^{\mathsf{real}})\not\equiv 0$, and thus step 3 returns "$e\neq 0$".

Now assume that every full α-monomial $\widetilde{\alpha}_s$ satisfies $\widetilde{\alpha}_s(\mathcal{V}^{\mathsf{real}})\equiv 0$. Then Sim^0 reaches step 4 with probability 1. Notice that e' contains exactly the non-full α-monomials in e. We show in Lemma 4.16 that for any valid element e' containing no full α-monomials, e' computes the identically zero polynomial iff each of its α-monomials is 0 on $\mathcal{V}^{\mathsf{real}}$. Thus, with probability $1-\mathsf{negl}(n)$ over the randomness of Sim^0, step 4 returns "$e=0$" iff each α-monomial $\widetilde{\alpha}$ in e satisfies $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}})\equiv 0$. ☐

We now prove the lemmas used in Theorem 4.14. The first is [BGK+14, Claim 8]; for completeness we include a proof in Appendix A.

Lemma 4.15 ([BGK+14])**.** *For any element e, if $e(\mathcal{V}^{\mathsf{real}})\not\equiv 0$ then $\Pr_{v\leftarrow\mathcal{V}^{\mathsf{real}}}\left[e(v)=0\right]=\mathsf{negl}(n)$.*

The next lemma states that if e has no *full* α-monomials, then it is identically 0 as a formal polynomial iff each of its α-monomials is 0 on $\mathcal{V}^{\mathsf{real}}$.

Lemma 4.16. *For any element e with no full α-monomials, e is the identically zero polynomial iff $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}})\equiv 0$ for every α-monomial $\widetilde{\alpha}$ in e.*

Proof. The "only if" direction is clear. For the "if" direction, we show that for any individual non-full α-monomial $\widetilde{\alpha}$, $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}})\equiv 0$ iff $\widetilde{\alpha}$ is identically zero (i.e. if its coefficient is identically zero).

Fix any non-full α-monomial $\widetilde{\alpha}$. We first show that the marginal distribution on the variables of $\{\tilde{s},\tilde{t},\tilde{B}_{i,b_1,b_2}\mid i\in[n],b_1,b_2\in\{0,1\}\}$ that appear in $\widetilde{\alpha}$'s coefficient consists of uniform non-zero vectors and uniform non-singular matrices. Let $\mathcal{C}\subseteq\left\{C_{i,b_1^i,b_2^i}:\ i\in[n],b_1^i,b_2^i\in\{0,1\}\right\}$ denote the set of matrices from which $\widetilde{\alpha}$'s variables come. Notice that $\mathcal{C}$ contains at most one matrix from every layer of the RMBP, because if $C_{i,b_1,b_2}\in\mathcal{B}$ then α_{i,b_1,b_2} appears in the monomial $\widetilde{\alpha}$, but $\widetilde{\alpha}$ contains at most one α_{i,b_1,b_2} from every layer i. Let $\mathcal{I}\subset[n]$ denote the layers from which $\mathcal{C}$ contains a matrix, and let $\mathcal{B} = \left\{\tilde{B}_{i,b_1^i,b_2^i} : C_{i,b_1^i,b_2^i} \in \mathcal{C}\right\}$. Then the marginal distribution of $\mathcal{V}^{\mathsf{real}}$ on $\tilde{s}$, $\tilde{t}$, and $\mathcal{B}$ is

$$\begin{aligned}\tilde{s} &= e_1 \cdot R_0^{-1} \\ \tilde{B}_{i,b_1^i,b_2^i} &= R_{i-1} \cdot B_{i,b_1^i,b_2^i} \cdot R_i^{-1}, \quad \forall i \in \mathcal{I} \\ \tilde{t} &= R_n \cdot e_w\end{aligned}$$

where each $B_{i,b_1^i,b_2^i} \in \mathbb{Z}_p^{w\times w}$ is a fixed non-singular matrix, and each $R_i \in \mathbb{Z}_p^{w\times w}$ is a uniform non-singular matrix. As noted above, there is at most one $C_{i,b_1^i,b_2^i} \in \mathcal{C}$ for each $i \in \mathcal{I}$, i.e., at most one $\tilde{B}_{i,b_1^i,b_2^i}$ on which $\widetilde{\alpha}$ depends, for every $i \in \mathcal{I}$. Consequently, the random matrices $\{R_i \mid i = 0, \ldots, n\}$ can be assigned to these equations in a way so that at most one random matrix is assigned to each equation. (This is because $|\mathcal{I}| < n$ because $\widetilde{\alpha}$ is not a full monomial, and thus there are $\leq n+1$ equations and there are $n+1$ random matrices.) Thus, the left-hand side of each equation is uniform in its support, even conditioned on any fixing of the other left-hand sides. Since the supports are all non-singular matrices (or all non-zero vectors in the case of $\tilde{s}$ and $\tilde{t}$), we have that when restricted to these values, the distribution we have generated is identical to $\mathcal{V}^{\mathsf{real}}$.

Let $\mathcal{V}^{\mathsf{rand}}$ denote the distribution over assignments to the variables of $\widetilde{\alpha}$, when $\tilde{s}, \tilde{t}$ are replaced with uniform vectors u_s, u_t, and the matrices in $\mathcal{B}$ are replaced with uniformly random matrices $M_1, \ldots, M_{|\mathcal{I}|}$. Because $\Pr_{u \leftarrow \mathbb{Z}_p^w}[u \neq 0^w] = 1 - p^{-w} = 1 - \mathsf{negl}(n)$, and because a uniform matrix in $\mathbb{Z}_p^{w\times w}$ is non-singular with probability $\geq 1 - w/p = 1 - \mathsf{negl}(n)$, the distributions $\left\{u_s, u_t, M_1, \ldots, M_{|\mathcal{I}|}\right\}$ and $\left\{\tilde{s}, \tilde{t}, \tilde{B}_{i,b_1^i,b_2^i} \in \mathcal{B}\right\}$ are $\mathsf{negl}(n)$-close in statistical distance. Thus because applying a deterministic function to random variables does not increase the statistical distance, we have

$$\left| \Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}\left[\widetilde{\alpha}(v) = 0\right] - \Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}\left[\widetilde{\alpha}(v) = 0\right] \right| = \mathsf{negl}(n). \tag{1}$$

If $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}}) \not\equiv 0$ then clearly $\widetilde{\alpha}$ is not the zero polynomial. If on the other hand $\widetilde{\alpha}(\mathcal{V}^{\mathsf{real}}) \equiv 0$, then (1) implies $\Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}\left[\widetilde{\alpha}(v) = 0\right] = 1 - \mathsf{negl}(n)$. Thus because $\deg(\widetilde{\alpha}) < n$, the Schwartz-Zippel lemma implies that $\widetilde{\alpha}$ is the zero polynomial. $\square$

5 VBB security for unrestricted graded encodings

We now analyze the security of our construction against a polynomial-time adversary in an *unrestricted* graded encoding scheme. Recall that the difference from a multiplication-restricted graded encoding scheme is that the adversary is no longer required to only compute elements that correspond to a valid polynomial (in the sense of Def. 1.2). Thus in this setting, simulation of $+$, $-$, and $\times$ queries is trivial, since they are always allowed.

As before, the difficulty is in simulating zero-test queries. By definition in this model, elements e that correspond to invalid polynomials are always non-zero. Because we can efficiently test for validity as described in Section 4.1, we can restrict ourselves to valid elements e.

Throughout this section we let m_e denote the number of full α-monomials (in the sense of Def. 4.10) in a given element e. In Section 5.2 we give an algorithm for simulating zero-test queries that runs in time $\mathsf{poly}(m_e, |e|)$. Thus if every valid element e contains a polynomial number of full α-monomials, this algorithm is a VBB simulator (and in any case the algorithm gives an iO simulator).

In Section 5.1 we show that a bound on m_e follows from a new hypothesis which is closely related to a parameterized version of the Bounded Speedup Hypothesis introduced by Brakerski and Rothblum [BR14a, BR14b]. Thus under this new hypothesis we get a VBB simulator. However, we also observe that obtaining an unconditional polynomial bound on m_e would imply the algebraic analog of P $\neq$ NP, namely VP $\neq$ VNP.

Definition 5.1 (VP [Val79])**.** Let $m(n)$ be a polynomial. A family $\mathcal{F} = \left\{f_n : \{0,1\}^{m(n)} \to \{0,1\}\right\}_n$ is in VP if for every n, $\mathsf{deg}(f_n) = \mathsf{poly}(n)$, and there exists a polynomial $p(n)$, and a family $\{C_n\}_n$ of boolean circuits such that for any n, $|C_n| \le p(n)$ (where $|C|$ denotes the number of gates in C), and $C_n(x) = f_n(x)$ for every $x \in \{0,1\}^{m(n)}$.

Definition 5.2 (VNP [Val79])**.** Let $m(n), k(n)$ be polynomials. A family $\mathcal{F} = \left\{f_n : \{0,1\}^{m(n)} \to \{0,1\}\right\}_n$ is in VNP if there is a family $\mathcal{G} = \left\{g_n : \{0,1\}^{m(n)} \times \{0,1\}^{k(n)} \to \{0,1\}\right\}_n$ of functions such that $\mathcal{G} \in$ VP, and for every n, $f_n(x) = \sum_{y \in \{0,1\}^{k(n)}} g_n(x,y)$ for every $x \in \{0,1\}^{m(n)}$.

In the following theorem, the term "valid element" denotes an element which corresponds to a valid polynomial, and *is computable by a PPT algorithm.*

Theorem 5.3. *If $m_e \le \mathsf{poly}(|e|)$ for all valid elements e, then VP $\neq$ VNP.*

Proof of Theorem 5.3. This just follows from the fact that the polynomial containing *all* valid full α-monomials is in VNP, and thus showing it has no polynomial-size circuit implies VP $\neq$ VNP. $\square$

Remark 5.4. This theorem gives evidence that proving VBB security in the unrestricted model for any "natural" algebraic obfuscator will entail proving VP $\neq$ VNP. Indeed, all known algebraic obfuscators for an arbitrary function f construct, for each input x, a poly-size arithmetic circuit that evaluates to $f(x)$. Since any (even exponentially-large) sum of these circuits is a VNP function, VBB security in this setting would seem to entail VP $\neq$ VNP (assuming that an efficient simulator cannot fool every adversary that depends on an exponential number of outputs $f(x)$).

5.1 The p-Bounded Speedup Hypothesis

We first state our new hypothesis. It corresponds exactly to replacing 3SAT with Max-2-SAT in the Bounded Speedup Hypothesis [BR14a, BR14b], and adding a parameter determining the speedup quality. By Max-2-SAT, we refer to the decision version in which a 2CNF formula is in Max-2-SAT iff a 7/10 fraction of its clauses can be simultaneously satisfied. This problem is NP-complete by a standard reduction from 3SAT.

Definition 5.5 (X-Max-2-SAT solver)**.** Consider a set $X \subseteq \{0,1\}^n$. We say that an algorithm $\mathcal{A}$ is an *X-Max-2-SAT solver* if it solves the Max-2-SAT problem restricted to inputs in X. Namely given a 2CNF formula ϕ on n variables, $\mathcal{A}(\phi) = 1$ iff $\exists x \in X$ that satisfies a $\ge 7/10$ fraction of ϕ's clauses.

Assumption 5.6 (p-Bounded Speedup Hypothesis)**.** Let $p : \mathbb{N} \to \mathbb{N}$. Then for any X-Max-2-SAT solver that has size $t(n)$, $|X| \le p(\mathsf{poly}(t(n)))$.

The following proof is inspired by [BR14b, Lemma 3.14].

Lemma 5.7. *Assume the p-Bounded Speedup Hypothesis. Then for all valid elements e, $m_e \le p(\mathsf{poly}(|e|))$.*

Proof. Let $X \subseteq \{0,1\}^\ell$ be the set of input profiles corresponding to e's full α-monomials (thus $|X| = m_e$). We give an X-Max-2SAT solver that has size $\mathsf{poly}(|e|)$, and thus $m_e = p\left(|e|^{\omega(1)}\right)$ would contradict the p-Bounded Speedup Hypothesis.

Let a 2CNF formula $\phi : \{0,1\}^\ell \to \{0,1\}$ be given. We assume the following without loss of generality.

- ϕ contains at most $4\ell^2$ clauses (otherwise some are redundant and can be removed).
- In the obfuscated branching program over which e is defined, each pair of input bits is read in at least $4\ell^2$ different layers (we can add this many "dummy layers" to any BP).

- e consists of only full α-monomials. (If not, first extract the homogeneous degree-n part of e, which can be done in time $\mathsf{poly}(|e|)$ and has size $\mathsf{poly}(|e|)$ by Lemma 2.1. We also assume that e contains at least one such monomial, which we can check using Schwartz-Zippel.)

Fix some clause c in ϕ, and let (i, j) be the input bits read by c. We modify e so that the degree of each α-monomial whose profile satisfies c is reduced by 1. To do this, take any layer k reading (i, j) that has not been used before, and in e set every α_{k,b_1,b_2} to 1 except for the one that doesn't satisfy c. Note that we can always pick a layer we haven't used before because there are $\geq 4\ell^2$ for each pair (i, j). In the case that we have $i = j$, we instead take any unused layer k reading (i, i') for some $i' \neq i$, and set every α_{k,b_1,b_2} to 1 except for the (at most) two that don't satisfy c.

After doing this for each of the m clauses, we have that e contains a monomial of degree $\leq n - 7m/10$ iff some $x \in X$ satisfies $7m/10$ of ϕ's clauses. Let $e^{(d)}$ denote the homogeneous degree-d portion of e, and define $e' := \sum_{d=1}^{n-7m/10} e^{(d)}$ which can be computed in time $\mathsf{poly}(|e|)$ and has size $\mathsf{poly}(|e|)$ by Lemma 2.1. Then $e' \not\equiv 0$ iff some $x \in X$ satisfies $7m/10$ of ϕ's clauses. Using Schwartz-Zippel, we can test $e' \equiv 0$ up to an arbitrarily small error. Fixing the random coins and using the union bound gives an X-Max-2SAT solver that has size $\mathsf{poly}(|e|)$. $\square$

5.2 The Zero-Test Simulator

At a high level, the strategy for simulating zero-test queries is the same as in the previous section (Construction 4.13). First, we extract from an element e each of its full α-monomials. Then, we zero-test each full α-monomial individually by evaluating it on the reconstructed branching program, and we zero-test the remaining portion of e by checking if it is the identically zero polynomial. The zero-test then returns "$e = 0$" iff each of these zero-tests did as well.

Lemma 5.8 gives an algorithm that extracts the list of full α-monomials; a similar algorithm was used in the zero-testing procedure of [BR14b].

Lemma 5.8. *Let e be a circuit computing a valid polynomial that contains m_e full α-monomials. Then in time $\mathsf{poly}(|e|, m_e)$ one can produce a list $(s_1, \ldots, s_{m_e})$ of circuits such that s_i computes the ith full α-monomial and has size $\mathsf{poly}(|e|)$.*

Proof. First, replace e by its homogeneous degree-n part, which can be done in time $\mathsf{poly}(|e|)$ and has size $\mathsf{poly}(|e|)$ by Lemma 2.1. We define a recursive algorithm R that takes as input $(\alpha_1, \ldots, \alpha_k)$ for some $k \leq n$, where each $\alpha_i = \alpha_{i,b_1,b_2}$ for some $b_1, b_2 \in \{0, 1\}$. R returns a list of all α-monomials in e that contain $\prod_{i \leq k} \alpha_i$. Given such R, the list of all full α-monomials is given by $\bigcup_{b_1,b_2 \in \{0,1\}} R(\alpha_{1,b_1,b_2})$.

On input $(\alpha_1, \ldots, \alpha_k)$, if $k = n$ then we simply return $\prod_{i \leq n} \alpha_i$. Otherwise, let e' be the circuit obtained from e by setting to 0 each α_{i,b_1,b_2} that does not appear in R's input, for each $i \leq k$. Then, for each variable α_{k+1,b_1,b_2} in layer $k + 1$, check if it is present in any e' monomial using Lemma 2.3. For each α_{k+1,b_1,b_2} that passes this check, we recursively call $R(\alpha_1, \ldots, \alpha_k, \alpha_{k+1,b_1,b_2})$ and return the union of the ≤ 4 answers.

There is a 1-1 correspondence between the leaves of R's recursion tree and e's full α-monomials. Thus since the depth of each recursion is $n \leq |e|$ and since each step runs in $\mathsf{poly}(|e|)$-time, overall R runs in time $\mathsf{poly}(|e|, m_e)$. Finally, we note that for each α-monomial $\widetilde{\alpha}$ returned by R, we can extract it from e (including its coefficient) by setting to 0 all variables that do not appear in $\widetilde{\alpha}$. $\square$

After running this algorithm, we define the decomposition $D(e) := (s_1, \ldots, s_{m_e}, e - \sum_{i \leq m_e} s_i)$. Note that this satisfies the property that each element contains at most one full α-monomial. Then the remainder of the zero-test algorithm is identical to Construction 4.13. The proof of correctness is the same, and the algorithm can be shown to run in time $\mathsf{poly}(|e|, m_e)$ (in particular, using a polynomial p in the p-Bounded Speedup Hypothesis gives a VBB simulator). We omit further details, and this completes the proof of Theorem 1.7 and Corollary 1.8.

References

[AGIS14] Prabhanjan Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: avoiding Barrington's theorem. In *ACM Conference on Computer and Communcications Security (CCS)*, 2014.

[Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. of Computer and System Sciences*, 38(1):150–164, 1989.

[BBC$^+$14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *Theory of Cryptography*, pages 26–51. Springer, 2014.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 1–18, 2001.

[BGK$^+$14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 221–238, 2014.

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. http://eprint.iacr.org/.

[BR14a] Zvika Brakerski and Guy N Rothblum. Black-box obfuscation for d-CNFs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 235–250. ACM, 2014.

[BR14b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *11th Theory of Cryptography Conference TCC*, pages 1–25, 2014.

[Bür00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7. Springer, 2000.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. http://eprint.iacr.org/.

[CHL$^+$14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. http://eprint.iacr.org/. To appear at EUROCRYPT 2015.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *33rd Annual Cryptology Conference (CRYPTO)*, pages 476–493, 2013.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. http://eprint.iacr.org/.

[FMM12] Hervé Fournier, Guillaume Malod, and Stefan Mengel. Monomials in arithmetic circuits: Complete problems in the counting hierarchy. *Computational Complexity*, pages 1–30, 2012.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 1–17, 2013.

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 40–49, 2013.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. http://eprint.iacr.org/.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *6th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, pages 443–457, 2000.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *ACM Symp. on the Theory of Computing (STOC)*, pages 20–31, 1988.

[Val79] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

A Proof of Lemma 4.15

Proof of Lemma 4.15. As noted in [BGK+14], the claim would follow directly from the Schwartz-Zippel lemma, if $\mathcal{V}^{\mathsf{real}}$ had been uniformly distributed, or obtained from uniformly distributed variables by applying a low-degree polynomial. This is not the case for $\mathcal{V}^{\mathsf{real}}$ due to the dependency on the entries of the R_i's.

Syntactically, e is a polynomial in the entries of the matrices C_{i,b_1,b_2}, but as every entry of C_{i,b_1,b_2} is a multivariate polynomial in α_{i,b_1,b_2} and the entries of R_{i-1}, R_i^{-1}, we think of e as a polynomial in the variables $\alpha_{i,b_1,b_2}, R_i, R_i^{-1}$. (Note that elements from the fixed matrices B_{i,b_1,b_2} in the original branching program also appear in the polynomial.) We define a new polynomial p' as follows: $p' = e \cdot \Pi_{i\in[n]}\det(R_i)$. Then for every $v \in \mathcal{V}^{\mathsf{real}}$, $p'(v) = 0 \Leftrightarrow e(v) = 0$, because $R_0, ..., R_n$ are invertible.

Let m_k denote the k'th monomial in e. We define $\mathcal{I}_k \subseteq \{0, ..., n\}$ to be the set of indices such that m_k contains a variable corresponding to an entry of R_i^{-1}, and notice that for every $i \in \{0, 1, ..., n\}$, m_k contains *at most one* variable representing an entry of R_i^{-1}. Indeed, the entries of R_i^{-1} appear only in the entries of the matrices C_{i,b_1,b_2} of layer i, whose index-sets intersect, and consequently their entries cannot be multiplied. We define a new polynomial $\tilde{p}$ as follows. $\tilde{p}$ is obtained from e by replacing R_i^{-1} with the adjugate matrix $\mathsf{adj}(R_i)$, and multiplying every m_k by $\Pi_{i\notin\mathcal{I}_k}\det(R_i)$. Since $\mathsf{adj}(R) = R^{-1}\cdot\det(R)$ for every invertible matrix R, and every m_k contains at most one variable representing an entry of R_i^{-1}, then $\tilde{p}, p'$ are functionally equivalent.

Notice, however, that $\tilde{p}$ does not depend on variables representing the entries of the R_i^{-1}'s. Moreover, $\deg(\tilde{p}) = \mathsf{poly}(\deg(e)) = \mathsf{poly}(n)$, because $\mathsf{adj}(R)$ is computable from the entries of R by a polynomial of degree $\mathsf{poly}(w)$ (where w is the dimension of R). Let $\mathcal{V}^{\mathsf{rand}}$ denote the distribution over assignments to the variables of $\tilde{p}$, when $R_0, ..., R_n$ are replaced with uniformly random matrices $M_0, ..., M_n$. The random variables $(R_0, ..., R_n), (M_0, ..., M_n)$ are statistically close (as observed in the proof of Lemma 4.16), so $|\Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}[\tilde{p}(v) = 0] - \Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}[\tilde{p}(v) = 0]| = \mathsf{negl}(n)$ (because the statistical distance does not increase when a deterministic function is applied to the random variables). Moreover, since $\deg(\tilde{p}) = \mathsf{poly}(n)$, then $\Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}[\tilde{p}(v) = 0] = \mathsf{negl}(n)$ by the Schwartz-Zippel lemma. Consequently,

$$\Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}[e(v) = 0] = \Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}\left[p'(v) = 0\right] = \Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}[\tilde{p}(v) = 0] \leq$$

$$\leq \left|\Pr_{v \leftarrow \mathcal{V}^{\mathsf{real}}}[\tilde{p}(v) = 0] - \Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}[\tilde{p}(v) = 0]\right| + \left|\Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}[\tilde{p}(v) = 0]\right| = \mathsf{negl}(n)$$

□