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

November 23, 2015

Abstract

Obfuscation, the task of compiling circuits or programs to make the internal computation unintelligible while preserving input/output functionality, has become an object of central focus in the cryptographic community. A work of Garg et al. [FOCS 2013] gave the first candidate obfuscator for general polynomial-size circuits, and led to several other works constructing candidate obfuscators. Each of these constructions is built upon another cryptographic primitive called a multilinear map, or alternatively a graded encoding scheme.

Several of these candidates have been shown to achieve the strongest notion of security (virtual black-box, or VBB) against "purely algebraic" attacks in a model that we call the *fully-restricted* graded encoding model. In this model, each operation performed by an adversary is required to obey the algebraic restrictions of the graded encoding scheme. These restrictions essentially impose strong forms of homogeneity and multilinearity on the allowed polynomials. While important, the scope of the security proofs is limited by the stringency of these restrictions.

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 attacks than considered in previous works. We also explore connections between notions of obfuscation security and longstanding questions in arithmetic circuit complexity. Our results include the following.

- In the arithmetic setting where the adversary is limited to creating multilinear, but not necessarily homogenous polynomials, we obtain an unconditional proof of VBB security. This requires a substantially different analysis than previous security proofs.
- In the arithmetic setting where the adversary can create polynomials of arbitrary degree, we show that a proof of VBB security for any currently available candidate obfuscator would imply VP $\neq$ VNP. To complement this, we show that a weaker notion of security (indistinguishability obfuscation) can be achieved unconditionally in this setting, and that VBB security can be achieved under a complexity-theoretic assumption related to the ETH.

*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.

†Supported by ERC starting grant 259426.

1 Introduction

Obfuscation, the task of compiling circuits or programs to make the internal computation unintelligible while preserving input/output functionality, has become an object of central focus in the cryptographic community. The eventual goal of this research is to construct an obfuscation scheme and prove that no polynomial-size adversarial Boolean circuit can attack it (under plausible notions of security discussed below). However, we remain very far from being able to prove such results. In this work, we study relaxations of this problem, and give a clean algebraic construction for which we can provably rule out wide classes of attacks. Specifically, we consider attacks corresponding to adversaries that only utilize restricted, but natural, classes of arithmetic circuits. We also explore connections between notions of obfuscation security and longstanding questions in arithmetic circuit complexity.

Background. Obfuscation was first formalized in the work of Barak et al. [BGI+12] (see also Hada [Had00]), who showed that the strongest notion of security is impossible to achieve in general. This notion, called *virtual black-box* (VBB) security, requires that an adversary who sees an obfuscated program gains only negligible advantage in computing any predicate on that program, as compared to an adversary who has only black-box access. The impossibility result shows that there exist (contrived) polynomial-time programs for which VBB security cannot be achieved.

In addition to this result, [BGI+12] defined a weaker security notion called *indistinguishability obfuscation* (iO), for which no impossibility result is known. iO security instead requires only that an adversary cannot distinguish between obfuscations of any two functionally-equivalent programs. The work of Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b] gave the first construction of an obfuscator for general polynomial-time programs that is a candidate for achieving iO security. This led to several following works constructing candidate obfuscators [BR14b, BGK+14, AGIS14, Zim15, AB15, BMSZ15], with improvements in both efficiency and security analysis.[1] We note that the set of programs to which the VBB impossibility result applies is not completely understood, and these obfuscators are also candidates for achieving VBB security whenever it is possible.

Current candidate obfuscators all rely on a cryptographic primitive called a *multilinear map* or alternatively a *graded encoding scheme* (GES); we use the latter term. A GES allows plaintext elements to be encoded at certain "levels", and allows algebraic operations on the encodings subject to restrictions on these levels. (For example, one common restriction allows encodings to be added or subtracted only if they are at the same level.) A GES also provides a public parameter that allows for encodings at the "top" level to be zero-tested, which reveals whether the underlying plaintext is 0. There are candidate GES constructions [GGH13a, CLT13, GGH14], and while attacks on these schemes have been found in some settings [CHL+15, BWZ14, CGH+15, HJ15, BGH+15, Hal15], no attacks are known on their use in any candidate obfuscator.

Each of the available candidate obfuscators belongs to a category that we call *algebraic obfuscators*. An algebraic obfuscator takes as input a function $f : \{0,1\}^n \to \{0,1\}$ (represented as a $\mathsf{poly}(n)$-size circuit, say), and outputs a $\mathsf{poly}(n)$-size set of encodings $\{k_i\}_i$ in some GES. The obfuscator also provides an evaluation circuit E such that, for every $x \in \{0,1\}^n$, the computation of $E(\{k_i\}_i, x)$ obeys the algebraic restrictions of the GES, and outputs a top-level encoding that encodes 0 iff $f(x) = 0$. Thus, in combination with the GES's zero-testing parameter, E and $\{k_i\}_i$ allow f to be evaluated on any input.

The primary security analyses available for algebraic obfuscators are in a generic model that we refer to as the *fully-restricted* GES model. In this model, each operation performed by an adversary

[1] [GGH+13b], along with a work of Sahai and Waters [SW14], has also led to a large body of research on how iO can be *used* to solve a variety of cryptographic problems. However, this will not be our focus here.

is required to obey the algebraic restrictions of the GES, which is formalized in the standard way by giving "pointers" to the encoded elements and only permitting certain operations on these pointers. Several of the candidate obfuscators are shown to unconditionally achieve VBB security in this model[2], which also implies iO security. The model is motivated by the plausible assumption that no useful information can be extracted from malformed encodings; indeed, even the known GES attacks all obey the algebraic restrictions. (These attacks exploit extra information that is given by the zero-test implementation, and require a structured set of 0-encodings that candidate obfuscators do not appear to permit.)

Algebraic security. As discussed in detail below, the task of proving security in an algebraic model is dominated by simulating the zero-test procedure. For this, given an arithmetic circuit C defined over the obfuscation $\{k_i\}_i$, one must decide whether $C(\{k_i\}_i) = 0$ (or more precisely, whether C outputs an encoding of 0), using only black-box access to the function f being obfuscated. Even iO security can be characterized this way, though in this case the simulator is not required to run in polynomial time [GGH+13b]. Currently available security proofs crucially rely on the fact that *every gate* in the circuit is required to compute a polynomial that obeys the GES's algebraic restrictions.

The scope of these security proofs is limited by the fact that these algebraic restrictions are quite strong. Though the restrictions are motivated by current GES constructions, and specifically the assumption that malformed encodings do not reveal useful information, it would be far preferable to prove security while allowing a broader set of algebraic operations. Further, understanding the extent to which the restrictions can be relaxed while still preserving security is a natural question.

In particular, if we delve deeper into existing GES constructions, there are several *purely algebraic operations* that an adversary can perform that do not correspond to permitted algebraic operations in the fully-restricted GES model. For example, in [GGH13a, CLT13], all encodings, regardless of their level, are represented as elements of a single ring. Thus, encodings at disparate levels can be added using an arithmetic operation, and yet this is not captured in the fully-restricted GES model.

Indeed, though the assumption that malformed encodings do not reveal useful information may be plausible, this does not justify the requirement that all *intermediate* steps an adversary may take must produce valid encodings. That is, there may well be a poly-size arithmetic circuit whose formal polynomial over the encodings evaluates to a valid top-level encoding (which could then be used in an attack), and yet this same polynomial is not computable by any poly-size circuit whose intermediate gates all obey the restrictions. Previous models do not capture such attacks, while ours do.

This is analogous to the situation with multilinear polynomials in the classical arithmetic setting. Namely, the best known transformation [NW97, Lem. 2] of a general circuit computing a multilinear polynomial into one in which *every gate* computes a multilinear polynomial incurs an exponential blowup in size. As described in more detail below, the GES's algebraic restrictions in fact impose a strong form of multilinearity, so the comparison is particularly apt.

Interestingly, we show that these restrictions are also related to other fundamental questions in arithmetic circuit complexity. Specifically, one of our results below shows that proving VBB security for an algebraic obfuscator in the most general GES model would imply the algebraic analog of P $\neq$ NP, namely VP $\neq$ VNP. In contrast, we give a construction for which iO security in this model can be proved unconditionally.

Finally, if we are able to deal with broader classes of algebraic attacks, this not only furthers our understanding of implementations using current GES constructions, but also allows for greater compatibility with potential future constructions. For example, in future constructions it may be possible that elements can be zero-tested at any level of the encoding. Our new models allow the adversary this flexibility, while in all other works this is not permitted.

[2]Note that this does not contradict [BGI+12] because the adversary is restricted.

1.1 Our results

We give a new candidate obfuscator for NC^1 circuits[3], and we 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 (as mentioned above) by actual attack scenarios that could arise using current GES constructions.

To set up our results, we first describe the restrictions imposed by the GES model used in previous works, and how our models differ. Throughout this section, we let $\mathcal{R}$ denote a commutative ring, and $\mathbb{U}$ denote a universe set. In a GES, each encoding of a plaintext $r \in \mathcal{R}$ is assigned an "index-set" $S \subseteq \mathbb{U}$; we denote such an encoding by $[r]_S$.

In the GES model considered by previous works, there are three restrictions. First, two encodings can be added or subtracted only if they have the same index-set. Second, two encodings can be multiplied only if their index-sets are disjoint. Third, only encodings whose index-set is the universe $\mathbb{U}$ can be zero-tested. Formally, we have the following.

Definition 1.1 (Fully restricted GES)**.** A *fully-restricted graded encoding scheme (GES)* consists of a set of basic elements of the form $[r]_S$ where $r \in \mathcal{R}$ is called the *value* and $S \subseteq \mathbb{U}$ is called the *index set*; three operations $+$, $-$, and $\times$; and a predicate IsZero.

An arithmetic operation $\circ$ on a pair $[r]_S, [r']_{S'}$, if defined, returns the element $[r \circ r']_{S \cup S'}$. The operations $+, -$ are defined only if $S = S'$, and $\times$ is defined only if $S \cap S' = \emptyset$. Finally, $\mathsf{IsZero}([r]_S) =$ True iff $r = 0$ and $S = \mathbb{U}$.

In discussing our new models, we will use the following notion of a *graded-multilinear* polynomial. (This is similar to the notion of a *set-multilinear* polynomial; cf. [NW97, FLMS14].)

Definition 1.2. We say that a polynomial p over a set $\{[r_i]_{S_i}\}_i$ is *graded-multilinear* if it is multilinear, and for every $[r_i]_{S_i}, [r_j]_{S_j}$ that appear together in a monomial of p, $S_i \cap S_j = \emptyset$.

Notice that in a fully-restricted GES, any element can be viewed as an arithmetic circuit over the basic elements, where each gate in the circuit is required to compute a graded-multilinear polynomial whose monomials all have the same index-set (the index-set of monomial $[r_1]_{S_1} \cdots [r_m]_{S_m}$ is defined as $\bigcup_{i \le m} S_i$). In our two new models, we relax these requirements on the intermediate gates. Specifically, in the first new model (Def. 1.3) we do not require that all monomials have the same index-set, though each intermediate gate is still required to compute a graded-multilinear polynomial. In the second new model (Def. 1.4), we allow the intermediate gates to compute any polynomial, though zero-testing is still only meaningful for elements computed by graded-multilinear polynomials. Formally, we have the following.

Definition 1.3 (Multiplication-restricted GES)**.** A *multiplication-restricted graded encoding scheme (GES)* consists of a set of basic elements $\{[r_i]_{S_i}\}_i$; formal arithmetic expressions defined over them (where basic elements are viewed as expressions of size 1); and a predicate IsZero.

An arithmetic operation $\circ \in \{+, -, \times\}$ on a pair of expressions e_1, e_2, if defined, outputs the formal expression $e_1 \circ e_2$. The $+, -$ operations are always defined, whereas $\times$ is defined only if $e_1 \times e_2$ computes a graded-multilinear polynomial. $\mathsf{IsZero}(e)$ returns True iff e computes a graded-multilinear polynomial whose evaluation on the basic elements has value $0 \in \mathcal{R}$.

Definition 1.4 (Unrestricted GES)**.** An *unrestricted graded encoding scheme (GES)* consists of a set of basic elements $\{[r_i]_{S_i}\}_i$; formal arithmetic expressions defined over them (where basic elements are viewed as expressions of size 1); and a predicate IsZero.

[3]Thanks to known bootstrapping theorems [GGH+13b, BR14b, App14], proving VBB (or iO) security for NC^1 implies the same for all polynomial-size circuits, under standard cryptographic assumptions.

An arithmetic operation $\circ \in \{+, -, \times\}$ on a pair of expressions e_1, e_2 is always defined, and outputs the formal expression $e_1 \circ e_2$. $\mathsf{IsZero}(e)$ returns True iff e computes a graded-multilinear polynomial whose evaluation on the basic elements has value $0 \in \mathcal{R}$.

Remark 1.5. Another sensible criterion for evaluating IsZero, inspired by current GES constructions, is the following: $\mathsf{IsZero}(e)$ = True iff e is a graded-multilinear polynomial that evaluates to 0, and further each of its monomials has index set $\mathbb{U}$. (This mimics the zero-testing requirement in a fully-restricted GES.) Our results below hold for this more restricted variant as well.

We now state our results. We defer until Section 2 the formal definition of VBB/iO security and an "ideal" GES. The latter is a standard formalization of an adversary that is restricted to the defined set of arithmetic operations, i.e. that cannot use features of the encodings' representation.

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

Theorem 1.6. *There exists a polynomial-time obfuscator that achieves VBB security for all* NC^1 *circuits in the multiplication-restricted ideal GES model.*

Turning to the unrestricted GES setting, we show that constructing an algebraic obfuscator that achieves VBB security would imply VP $\neq$ VNP. (Recall from above that the notion of an algebraic obfuscator captures all candidates currently available; see Definition 4.3 for a formal definition.)

Theorem 1.7. *If there exists a polynomial-time algebraic obfuscator that achieves VBB security for all* NC^1 *circuits in the unrestricted ideal GES model, then* VP $\neq$ VNP.

We complement this with two other results. First, we show that iO security can be achieved unconditionally in the unrestricted setting.

Theorem 1.8. *There exists a polynomial-time obfuscator that achieves iO security for all* NC^1 *circuits in the unrestricted ideal GES model.*

Second, we show that VBB security in this setting can be achieved under a complexity-theoretic assumption (related to one used in [BR14a, BR14b]). We defer the details of this assumption until Section 4.1, but it is essentially a parameterized strengthening of the Exponential Time Hypothesis.

Theorem 1.9. *Assume the p-Bounded Speedup Hypothesis for some function $p : \mathbb{N} \to \mathbb{N}$. Then there is a polynomial-time obfuscator that achieves VBB security for all* NC^1 *circuits in the unrestricted ideal GES model, where the simulator against time-t adversaries runs in time $p(t^{O(1)})$.*

1.2 Techniques

Our obfuscator in Theorems 1.6, 1.8, and 1.9 is a modified version of the construction due to Ananth et al. [AGIS14], with the primary difference being that we use a stronger notion of "straddling sets"; these are defined formally in Section 2.2 and sketched below. Before outlining our proofs, we review this construction (which in fact is slightly more involved than described here).

Obfuscating the function. For a given NC^1 function $f : \{0,1\}^\ell \to \{0,1\}$, we first construct a length-n, width-w oblivious matrix branching program (BP) over $\mathbb{Z}_p$ for a prime $p = 2^{\Omega(\ell)}$. Recall that this is a set $\{B_{i,b} \mid i \in [n], b \in \{0,1\}\}$ of $w \times w$ matrices satisfying $B_x[1,w] = 0 \Leftrightarrow f(x) = 0$, where $B_x := \prod_{i=1}^{n} B_{i,x_{\mathsf{inp}(i)}}$ and $\mathsf{inp} : [n] \to [\ell]$ specifies the input bit read in each layer. That is, the value of $f(x)$ is encoded by the top-right entry of B_x. We note that n, w, and inp depend only on

the input length ℓ. For this overview one can think of the BP being constructed via Barrington's theorem [Bar89], though [AGIS14] in fact uses a more efficient construction.

The BP is then randomized in two steps, the first of which is Kilian's technique [Kil88] (cf. [Bab87]). Recall that for this we choose $n-1$ non-singular matrices $R_1, \ldots, R_{n-1} \in \mathbb{Z}_p^{w \times w}$ 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 (where $R_0 = R_n = I_{w \times w}$). Next, we further randomize by choosing $2n$ uniform and independent scalars $\alpha_{i,b} \in \mathbb{Z}_p \setminus \{0\}$, and set $C_{i,b} := \alpha_{i,b} \cdot \widetilde{B}_{i,b}$. It can be easily verified that the new BP $\{C_{i,b} \mid i, b\}$ computes the same function as before with probability 1. Following [Kil88], it can be shown 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 sampled given only x and $f(x)$.

The final step in the obfuscation is to encode the matrix elements using the GES. 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 GES 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 ideal GES 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 create a unique formal variable for each entry of each matrix $C_{i,b}$, assign to each variable the corresponding index-set used by the obfuscator, and give these to the adversary. (Up to now the simulation is perfect, because in an ideal GES the adversary sees only random representations of the encoded elements.)

In all prior works, simulating the arithmetic operations $+, -, \times$ was trivial, because in a fully-restricted GES 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 $\times$ in the multiplication-restricted setting involves checking whether a given arithmetic circuit computes a graded-multilinear polynomial. To solve this we show that checking for graded-multilinearity is reducible to identity-testing, so we can use the Schwartz-Zippel lemma to efficiently check up to a negligible error probability. Specifically, testing for multilinearity [FMM15] and testing whether two distinct variables appear together in a monomial (Lemma 2.4) both reduce to identity testing.

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 graded-multilinear polynomial over the initial elements, and needs to check if e evaluates to 0 on the obfuscated program.

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 $\mathbb{Z}_p \setminus \{0\}$, so we can 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 the independence of the $\alpha_{i,b}$ implies that, with high probability, e evaluates to 0 on the obfuscation iff each of its α-monomials do. Furthermore the marginal distribution on each α-monomial can be simulated with only black-box access to f, because each can contain at most one $\alpha_{i,b}$ for each layer i (due to the index-set conditions mentioned above). So, if we can efficiently decompose e into the formal sum of its α-monomials, and show that there are at most $\mathsf{poly}(n)$ of them, then the zero-test can be efficiently simulated.

The approach of viewing e in this way was used in the prior works [BR14b, BGK+14, AGIS14]. Specifically, [BR14b] gives a procedure for decomposing e into its α-monomials that runs in time proportional to the number of such monomials, and then shows that under a complexity-theoretic

assumption there are at most $\mathsf{poly}(n)$ of them. [BGK+14] gives a different procedure (also used by [AGIS14]) for decomposing e into its degree-n α-monomials, and then shows that if e was constructed using a fully-restricted GES, 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 complexity assumptions when possible. Second, there are elements e in a multiplication-restricted GES 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 by giving a new decomposition algorithm, and we show that it runs in polynomial time on any e constructed in a multiplication-restricted GES. Our decomposition algorithm differs from previous works in several ways. One high-level difference is that it does not isolate each α-monomial by itself, but rather returns a set of polynomials 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, as the decomposition proceeds downward, we use arithmetic circuit analysis tools to check whether the expression computed by a given gate contains any degree-n α-monomials; if not, we do not decompose it further. Further, for a multiplication gate whose expression does contain degree-n α-monomials, we show that at most one of its children requires further decomposition, and we again use circuit analysis tools to select the appropriate child. This crucially prevents the exponential blowup that would be incurred by the [BGK+14] decomposition in the multiplication-restricted setting. (Addition gates are easier to handle, because they can be directly absorbed into the decomposition, increasing the number of summands by at most the number of input wires.) Thus, we are able to efficiently decompose any graded-multilinear polynomial into a sum over $\mathsf{poly}(n)$ sub-polynomials, each of which contains at most a single degree-n α-monomial.

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, and zero-tested using Kilian's simulation as mentioned above. To complete the zero-testing algorithm, we show that the set of all α-monomials with degree $< n$ can be *collectively* zero-tested using Schwartz-Zippel, because with high probability each is zero on the obfuscation iff it is the identically zero polynomial. This completes the overview of Theorem 1.6.

Unrestricted GES. We now turn to the unrestricted ideal GES model. Here we no longer assume that every gate in an expression e computes a graded-multilinear polynomial, though we still only need to simulate the zero-test for expressions whose output gate does.

For the proof of Theorem 1.7, first observe that for any algebraic obfuscator that produces a set of encodings $\{k_i\}_i$ and an evaluation circuit E, the expression

$$g(\{k_i\}_i) := \sum_{x \in \{0,1\}^\ell} E(\{k_i\}_i, x)$$

computes a graded-multilinear polynomial and is in VNP. Now consider the distribution on $f : \{0,1\}^\ell \to \{0,1\}$ where f is identically 0 with probability 1/2, and otherwise f outputs 1 on only a single, uniform $x \in \{0,1\}^\ell$. Because g outputs (an encoding of) 0 or 1 respectively in these two cases, if VP = VNP then, under any unrestricted GES, there is a poly-time adversary that can perfectly distinguish them. However, these cases cannot be distinguished in poly-time with more than negligible probability using only black-box access to f, so VBB security implies VP $\neq$ VNP.

To prove Theorem 1.8, we use the fact that iO security is equivalent to VBB security with an unbounded-time simulator [GGH+13b]. Thus we can follow the outline of Theorem 1.6, but just

directly decompose any expression into its (possibly exponential-sized) sum of α-monomials.

To prove Theorem 1.9, we take the approach of Brakerski and Rothblum [BR14a, BR14b] and bound the number of degree-n α-monomials under a complexity-theoretic assumption related to the Exponential Time Hypothesis. With this bound, we apply essentially the algorithm from [BR14b] for zero-testing. One important difference is that here we cannot guarantee that e contains *only* degree-n 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 GES, and the proof of Theorem 1.6, appear in Section 3. In Section 4 we analyze the unrestricted GES and prove Theorems 1.7-1.9.

2 Preliminaries

Throughout, $\mathsf{poly}(n)$ refers to a function of n that is bounded above by n^c for some constant c and sufficiently large n, and $\mathsf{negl}(n)$ refers to a function of n that is bounded above by $1/n^c$ for every constant c and sufficiently large n.

2.1 Arithmetic circuit tools

We use several tools for analyzing and modifying arithmetic circuits. One is the classical algorithm for extracting the homogeneous degree-d portion of an arithmetic circuit, a proof of which 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.*

The following arithmetic circuit testing procedures are 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, so applying the Schwartz-Zippel lemma over a sufficiently large field gives a $\mathsf{poly}(n)$-time algorithm with error probability $\mathsf{negl}(n)$.

Lemma 2.2 (Multilinearity check; [FMM15, 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 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 monomials from e in which x appears. Then $e^{(x)} \equiv 0$ iff x appears in no monomial of e. □

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 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 monomial of e contains both x and x'. □

2.2 Strong straddling sets

Here we define the notion of *strong* straddling set systems, which strengthen the straddling set systems introduced by Barak et al. [BGK+14]. These set systems are used in choosing the index-sets for the graded encoding scheme, as described in Section 1.2.

In particular, the fact that $S_{i,0} \cap S_{j,1} \neq \emptyset$ for each $i, j \in [n]$ is used to ensure that no graded-multilinear monomial contains GES elements from matrices corresponding to both "$x_k = 0$" and "$x_k = 1$" for any $k \in [\ell]$, where $x \in \{0,1\}^\ell$ is the input to the function being obfuscated. (To see the intuition for this, think of i and j as two layers in the branching program that both read the same input bit.) We note however that the actual construction (Section 2.6) uses several straddling set systems to ensure that, e.g., elements corresponding to "$x_1 = 0$" and "$x_2 = 1$" *can* be multiplied (as is necessary for the evaluation of any input whose first two bits are 01).

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 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 for all $1 \leq i \leq n$,

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

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 GES replaced by our two new GESs (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 GES. $\mathcal{M}$ maintains a list of elements, and allows a user to perform 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 from all other handles (even if the same element appears multiple times in the table, distinct handles are used). The user sees only the handles, and queries $\mathcal{M}$ with them to evaluate the operations of the GES.

$\mathcal{M}$ 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 GES, 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

As in [AGIS14], our obfuscator first transforms the input circuit F into a *dual-input, oblivious, relaxed matrix branching program (RMBP)*, which will then be obfuscated. The differences from standard

branching programs are that (1) each layer reads two input bits, (2) which input bits are read in each layer depends only on the input length, and (3) the function's output is determined by just one entry of the product matrix.

Definition 2.7 (Dual-input RMBP)**.** Let $\mathcal{R}$ be any finite ring. A *dual-input relaxed matrix branching program* (over $\mathcal{R}$) of width w and length n for ℓ-bit inputs is given by $\mathsf{BP} = \big(\mathsf{inp}_1, \mathsf{inp}_2, \{B_{i,b_1,b_2}\}_{i\in[n],b_1,b_2\in\{0,1\}}\big)$, where each $B_{i,b_1,b_2} \in \mathcal{R}^{w\times w}$ is full-rank, and $\mathsf{inp}_1, \mathsf{inp}_2 : [n] \to [\ell]$ select the input bits to be read in each layer.

BP defines a function from $\{0,1\}^\ell$ to $\{0,1\}$ as follows: $\mathsf{BP}(x) = 1$ if and only if $\Big(\prod_{i=1}^{n} B_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\Big)[1,w] \neq 0$. 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 notions in an idealized model

We consider two obfuscation security notions: Virtual Black Box (VBB) and Indistinguishability Obfuscation (iO) in the $\mathcal{M}$-idealized model, where $\mathcal{M}$ is some oracle. (These definitions are taken almost verbatim from [BBC+14].) In this model, the adversary, the obfuscator, and the circuit output by the obfuscator have access to the oracle $\mathcal{M}$, but the function family that is being obfuscated does not have access to $\mathcal{M}$.

Definition 2.8 (VBB security)**.** For a randomized oracle $\mathcal{M}$, and a circuit class $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$, we say that a uniform PPT oracle machine $\mathcal{O}$ *achieves VBB security* for $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ in the $\mathcal{M}$-idealized model, if the following conditions are satisfied:

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

$$\Pr[\widetilde{C}^{\mathcal{M}}(x) \neq C(x) \,|\, \widetilde{C} \leftarrow \mathcal{O}^{\mathcal{M}}(C)] = 1$$

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

- Polynomial Slowdown: For every $\ell \in \mathbb{N}$ and $C \in \mathcal{C}_\ell$: $|\mathcal{O}^{\mathcal{M}}(C)| = \mathsf{poly}(|C|)$.

- 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 $C \in \mathcal{C}_\ell$:

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

  where the probabilities are over the randomness 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. This is equivalent to the more well-known definition that requires obfuscations of any two functionally-equivalent programs to be polynomial-time indistinguishable [GGH+13b].

Definition 2.9 (iO security)**.** For $\mathcal{M}$ and $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ as in Definition 2.8, we say that $\mathcal{O}$ *achieves iO security* for $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ in the $\mathcal{M}$-idealized model, if it satisfies the three properties from Definition 2.8, except that Sim may run for an arbitrary finite amount of time.

2.6 The obfuscator construction

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

Specifically, on input an NC^1 circuit $F : \{0,1\}^\ell \to \{0,1\}$, the obfuscator $\mathcal{O}$ first converts F into a length-n, width-w 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 $\mathsf{inp}_1, \mathsf{inp}_2 : [n] \to [\ell]$ select the input bits read in each layer, and each $B_{i,b_1,b_2} \in \{0,1\}^{w\times w}$ has full rank. The exact details of the construction are not required here, but we will use the following three properties.

1. No input bit is read twice in the same layer. Formally, $\mathsf{inp}_1(i) \neq \mathsf{inp}_2(i)$ for every $i \in [n]$.
2. Every pair of input bits are paired in some layer. Formally, for every distinct $j, k \in [\ell]$, there exists $i \in [n]$ such that either $\mathsf{inp}_1(i) = j \wedge \mathsf{inp}_2(i) = k$ or $\mathsf{inp}_1(i) = k \wedge \mathsf{inp}_2(i) = j$.
3. There exists $\ell' \leq n$ such that every input bit is read in exactly ℓ' layers. Formally, $|\mathsf{ind}(j)| = \ell'$ for every $j \in [\ell]$, where $\mathsf{ind}(j) := \{i \in [n] : \ \mathsf{inp}_1(i) = j \ \vee \ \mathsf{inp}_2(i) = j\}$.

Randomizing BP. $\mathcal{O}$ samples a large enough $\Omega(n)$-bit prime p, and randomizes BP following [AGIS14, Sec. 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 consists of ideal encodings of the entries of $\tilde{s}, \tilde{t}$ and each C_{i,b_1,b_2}, as follows.

Encoding the randomized BP. Let $\mathbb{U}_s, \mathbb{U}_t, \mathbb{U}_1, ..., \mathbb{U}_\ell$ be disjoint sets such that $|\mathbb{U}_s| = |\mathbb{U}_t| = 1$, and $|\mathbb{U}_j| = 2\ell' - 1$ for every $j \in [\ell]$. Define $\mathbb{U} := U_s \cup U_t \cup \bigcup_{j\in[\ell]} \mathbb{U}_j$.

For $j \in [\ell]$, let $\mathbb{S}^j$ be a strong straddling set system with ℓ' entries over universe $\mathbb{U}_j$ (see Def. 2.5). 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: $\mathbb{S}^j = \left\{S^j_{k,b} : \ k \in \mathsf{ind}(j), b \in \{0,1\}\right\}$. For each $i \in [n]$ and $b_1, b_2 \in \{0,1\}$, we encode each entry of each matrix C_{i,b_1,b_2} with the set

$$S(i, b_1, b_2) := S^{\mathsf{inp}_1(i)}_{i,b_1} \cup S^{\mathsf{inp}_2(i)}_{i,b_2}.$$

$\mathbb{U}_s$ and $\mathbb{U}_t$ are used to encode the entries of $\tilde{s}, \tilde{t}$, respectively.

Formally, $\mathcal{O}$ initializes the oracle $\mathcal{M}$ with the ring $\mathbb{Z}_p$, the universe set $\mathbb{U}$, and the following set of basic elements:

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

As in the introduction, we use $[x]_S$ to denote that x is encoded with the index-set S. $\mathcal{M}$ returns handles for these elements, and $\mathcal{O}$ outputs these handles as the obfuscation of F. (We note that the vectors $\tilde{s}, \tilde{t}$ were omitted from the technical overview in Section 1.2, but they are easily incorporated into the analysis.)

3 VBB security for multiplication-restricted graded encodings

Let $\mathcal{O}$ denote the obfuscator from Section 2.6 when the oracle $\mathcal{M}$ is instantiated with the rules of a multiplication-restricted GES (Def. 1.3). In this section we construct an efficient simulator Sim that, given black-box access to a function $F : \{0,1\}^\ell \to \{0,1\}$, simulates the view of $\mathcal{A}^{\mathcal{M}}(\mathcal{O}^{\mathcal{M}}(F))$ for any polynomial-time adversary $\mathcal{A}$. This will prove Theorem 1.6.

Theorem 1.6. *There exists a polynomial-time obfuscator that achieves VBB security for all* NC^1 *circuits in the multiplication-restricted ideal GES model.*

Sim is given $1^{|F|}$, 1^ℓ, and a description of the adversary $\mathcal{A}$, and has oracle access to F. Sim first generates formal variables representing each entry of each matrix C_{i,b_1,b_2} and of the vectors $\tilde{s}$ and $\tilde{t}$, and simulates $\mathcal{M}$'s initialization by generating handles corresponding to these variables (using the same index sets as $\mathcal{O}$). 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 and zero-test queries.

3.1 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 graded-multilinear polynomial. Recall (Def. 1.2) that a polynomial is graded-multilinear if it is multilinear, and further the variables appearing in any single monomial have pairwise disjoint index sets.

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

If the query is valid, then Sim generates a new handle h for e, adds it to the handle set, and returns h to $\mathcal{A}$. The proof of the next lemma is immediate given Lemmas 2.2 and 2.4.

Lemma 3.1. *For every multiplication query* $e_1 \times e_2$, Sim *generates in* $\mathsf{poly}(n)$*-time an answer whose distribution is* $(1 - \mathsf{negl}(n))$*-close to that of* $\mathcal{M}$*'s answer.*

3.2 Simulating zero-test queries

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

Definition 3.2 (Touching matrices and layers)**.** We say that e *touches a matrix* C_{i,b_1,b_2}, for $i \in [n]$ and $b_1, b_2 \in \{0,1\}$, if some monomial in (the polynomial computed by) 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 *input profile* of an arithmetic circuit e, which represents the partial information that e gives about an input $x \in \{0,1\}^\ell$ to the obfuscated function. The input profile is a string in $\{0,1,*\}^\ell$ whose jth entry indicates whether e touches a matrix that corresponds to "$x_j = 0$", one that corresponds to "$x_j = 1$", or neither. (Circuits that touch conflicting matrices have profile $\perp$.) We also define *single-input* circuits as those whose variables all correspond to a single input x, i.e. whose input profile is not $\perp$.

Definition 3.3 (Input-profiles and single-input elements)**.** The *input-profile* $\mathsf{Prof}(e) \in \{0,1,*\}^{\ell} \cup \{\perp\}$ of a circuit e 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, and set $\mathsf{Prof}(e) = \perp$; in this case we say $\mathsf{Prof}(e)$ is *invalid*.

We say e is *single-input* if $\mathsf{Prof}(e) \neq \perp$, and that it has a *complete* profile if $\mathsf{Prof}(e) \in \{0,1\}^{\ell}$. We say e and e' *conflict on index* j if $\mathsf{Prof}(e)_j = 1 - \mathsf{Prof}(e')_j$, or if 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.

The simulator answers a zero-test query "$e = 0$?" as follows. First, it decomposes e into a list $\{e_1, ..., e_k\}$ of elements, such that: $e = \sum_{i=1}^{k} e_i$, with equivalence as polynomials; each e_i is either a single-input element or does not touch all layers $i \in [n]$; and $k = \mathsf{poly}(n)$. Then, the simulator extracts the full α-monomials (i.e. the monomials with degree n) from the single-input elements, and performs a zero-test on each separately. Finally, it performs a zero-test on the sum of non-full α-monomials (i.e. the monomials with degree $< n$). Next, we give the decomposition algorithm.

3.2.1 Decomposition algorithm

We give a decomposition $D(e)$ of a circuit e, satisfying the three properties in Figure 1.

1. $e = \sum_{s \in D(e)} s$, with equivalence as polynomials.
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 3.4. *For any circuit e whose gates each compute a graded-multilinear polynomial, there exists a $\mathsf{poly}(|e|)$-time algorithm that, with probability $1 - \mathsf{negl}(|e|)$, outputs a decomposition $D(e)$ satisfying the properties in Figure 1.*

For the decomposition, we view e as a layered, unbounded fan-in circuit whose layers alternate between addition (or subtraction) and multiplication gates. We further assume that all input wires to a layer come from the layer directly below, and that the top layer is a multiplication gate. 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 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. We let m denote the number of sections in e, and we number the sections starting from 1 at the top, so the m^{th} section contains e's input variables.

1. $e = \sum_{(z,z')\in \text{GO}} zz' + \sum_{(z,z')\in \text{STOP}} zz'$.
2. For each $(z, z') \in$ STOP, zz' computes a graded-multilinear polynomial that is either single-input or does not touch every layer.
3. After step i, for each $(z, z') \in$ GO: z is single-input, z' is a gate in section i, and zz' computes a graded-multilinear polynomial 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, |STOP| increases by at most the number of wires leaving section i.

Figure 2: Invariants of the decomposition algorithm

Lemma 3.5. *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 3.7 below. Thus, after step m GO cannot contain any (z, z') that satisfies the third invariant, so GO $= \emptyset$. □

Next, we prove Theorem 3.4. The proof uses a few simple lemmas that are proved afterwards. We will implicitly use the fact that the set of layers touched by any gate in e, and thus its input profile, can be efficiently computed (up to a $1 - \mathsf{negl}(|e|)$ error) using the algorithm from Lemma 2.3.

Proof of Theorem 3.4. 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 $= \emptyset$.

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 $(z, z') \in$ 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 3.8 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 $(z, q_1 \times \ldots \times q_k) \in$ GO with $(z \times \prod_{j\neq j^*} q_j,\ q_{j^*})$. By Lemma 3.7, 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 $(z, z') \in$ GO for each gate z' in section i, we repeatedly replace any $(z_1, z'), (z_2, z') \in$ GO with $(z_1 + z_2, z')$. Lemma 3.9 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 $(z, z') \in \text{GO}$ 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 3.2), thus ensuring that zq_j is graded-multilinear for each $j \leq k$. Then for each such $(z, q_1 + \cdots + q_k)$, we remove it from GO and set

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

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

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 $(z_1, z'), (z_2, z') \in \text{GO}$ with $(z_1 + z_2, z')$. □

Next, we prove a useful structural result on multilinear polynomials, and then prove the lemmas that were used in Theorem 3.4. We use $\mathcal{V}(e)$ to denote the set of variables that appear in the (non-zero) monomials of the polynomial computed by e; note that this may be a strict subset of the variables at e's input gates.

Lemma 3.6. *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

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

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

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

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 3.7. *If e_1 and e_2 are graded-multilinear and single-input, and $e_1 \times e_2$ is graded-multilinear, then $e_1 \times e_2$ is single-input.*

Proof. If $e_1 \times e_2$ conflicts on some index j, then there are variables $x_1 \in \mathcal{V}(e_1)$ and $x_2 \in \mathcal{V}(e_2)$ that conflict on index j, and cannot be multiplied. But by Lemma 3.6, x_1 and x_2 appear together in some monomial, so $e_1 \times e_2$ is not graded-multilinear. □

Lemma 3.8. *Let $e_1, \ldots, e_d$ be graded-multilinear such that $\prod_{i \leq d} e_i$ is graded-multilinear, 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 3.6, because any two variables from the same layer cannot be multiplied due to the index-set construction.

Pick any i such that e_i is not single-input (there must be one by Lemma 3.7). 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 3.9. *Assume $z_1 \times z'$ and $z_2 \times z'$ are graded-multilinear, 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 3.8, 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 not graded-multilinear. □

3.2.2 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 3.10. 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 graded-multilinear 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 3.11. *Let e be an arithmetic circuit whose gates each compute a graded-multilinear polynomial and let $D(e)$ be its decomposition given by Theorem 3.4. 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 due to the index-set construction, the only possible monomials of degree n are the full α-monomials. Further, we show in Lemma 3.16 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 2.6.)

Theorem 3.12 ([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 $x \in \{0,1\}^\ell$, $\{\mathsf{BP}'|_x\} \equiv \{\mathsf{Sim}'(1^{|F|}, F(x))\}$.*

We are now ready to describe the simulator.

Construction 3.13 (Zero-test simulator)**.** The zero-test simulator Sim^0 uses the decomposition algorithm D of Theorem 3.4. On input 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 $\mathsf{Sim}'\left(1^{|C|}, C\left(x\right)\right)$, 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 3.13 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.

Theorem 3.14. *Let e be an arithmetic circuit whose gates each compute a graded-multilinear polynomial, and let Sim^0 be as in Construction 3.13. 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 3.15 shows that for any such 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 3.13 returns "$e \neq 0$". Indeed, because the call to Sim' generates exactly the marginal distribution on $\widetilde{\alpha}_s$'s variables by Theorem 3.12, the evaluation generates a sample from $\widetilde{\alpha}_s(\mathcal{V}^{\mathsf{real}})$. By Lemma 3.15 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 3.16 that 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 state the lemmas used in Theorem 3.14. The first is [BGK+14, Claim 8]; for completeness we include a proof in Appendix A.

Lemma 3.15 ([BGK+14])**.** *For any valid 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 3.16. *If e contains no full α-monomials, then it 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 $M_1, ..., M_{|\mathcal{I}|}$. 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 $\left\{u_s, u_t, M_1, ..., M_{|\mathcal{I}|}\right\}$ 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

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

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 $\Pr_{v \leftarrow \mathcal{V}^{\mathsf{rand}}}[\widetilde{\alpha}(v) = 0] = 1 - \mathsf{negl}(n)$. Thus because $\deg(\widetilde{\alpha}) < n$, the Schwartz-Zippel lemma implies that $\widetilde{\alpha}$ is the zero polynomial. □

4 Security for unrestricted graded encoding schemes

We now analyze the security of our construction against a polynomial-time adversary in an *unrestricted* GES. Recall that the difference from a multiplication-restricted GES is that the adversary is no longer required to construct circuits in which every gate computes a graded-multilinear polynomial. Thus in this setting, simulating $+$, $-$, 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, IsZero only returns True on circuits that compute graded-multilinear polynomials. Because we can efficiently test for graded-multilinearity as described in Section 3.1, we can restrict ourselves to such circuits.

Throughout this section we let m_e denote the number of full α-monomials (in the sense of Def. 3.10) for a given circuit e. In Section 4.2 we give an algorithm for simulating zero-test queries that runs in time $\mathsf{poly}(m_e, |e|)$. Thus if every graded-multilinear polynomial computed by a polynomial-size circuit e contains a polynomial number of full α-monomials, this algorithm gives a VBB simulator (and, in any case, the algorithm gives an iO simulator).

In Section 4.1 we show that a polynomial 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] (see that section for further discussion). Thus under this new hypothesis we can achieve VBB security.

However, we first show that unconditionally obtaining a VBB simulator in this model for any algebraic obfuscator would imply the algebraic analog of P $\neq$ NP, namely VP $\neq$ VNP. Intuitively, this is because the polynomial summing a function's outputs over all possible inputs is in VNP, but this polynomial cannot in general be simulated with only black-box access. This is formalized in the proof of Theorem 1.7 below; we first state some definitions.

Definition 4.1 (VP [Val79])**.** Let $m(n) = \mathsf{poly}(n)$. 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 arithmetic circuits such that for any n, $|C_n| \leq p(n)$, and $C_n(x) = f_n(x)$ for every $x \in \{0,1\}^{m(n)}$.

Definition 4.2 (VNP [Val79])**.** Let $m(n), k(n) = \mathsf{poly}(n)$. A family $\mathcal{F} = \left\{f_n : \{0,1\}^{m(n)} \to \{0,1\}\right\}_n$ is in VNP if there exists a family $\mathcal{G} = \left\{g_n : \{0,1\}^{m(n)+k(n)} \to \{0,1\}\right\}_n$ in VP such that, for every n and every $x \in \{0,1\}^{m(n)}$, $f_n(x) = \sum_{y \in \{0,1\}^{k(n)}} g_n(x,y)$.

Definition 4.3 (Algebraic obfuscator)**.** An algorithm $\mathcal{O}$ that takes as input a circuit $f : \{0,1\}^n \to \{0,1\}$ is an *algebraic obfuscator* if $\mathcal{O}(f) = (\{k_i\}_i, E)$, where (1) $\{k_i\}_i$ is a $\mathsf{poly}(|f|)$-size set of encodings in some GES, (2) E is a $\mathsf{poly}(|f|)$-size arithmetic circuit, and (3) for every $x \in \{0,1\}^n$, the computation of $E(\{k_i\}_i, x)$ obeys the restrictions of the GES and satisfies $\mathsf{IsZero}(E(\{k_i\}_i, x)) =$ True iff $f(x) = 0$.

Theorem 1.7. *If there exists a polynomial-time algebraic obfuscator that achieves VBB security for all* NC^1 *circuits in the unrestricted ideal GES model, then* VP $\neq$ VNP.

Proof. Assume the existence of an obfuscator $\mathcal{O}$ as in the theorem statement, that on input a circuit to be obfuscated outputs an evaluation circuit E, and a set $\{k_i\}_i$ of encodings (namely, $\mathcal{O}$ instantiates the unrestricted ideal GES oracle with these encodings). We use $\mathcal{O}$ to construct a function family $\mathcal{F} \in \mathrm{VNP} \setminus \mathrm{VP}$. For every $n \in \mathbb{N}$, let $C_n : \{0,1\}^n \to \{0,1\}$ be the circuit that always outputs 0; and for every $x_0^n \in \{0,1\}^n$, let $C_{x_0^n} : \{0,1\}^n \to \{0,1\}$ be the circuit that outputs 1 only on x_0^n, i.e., $C_{x_0^n}(x_0^n) = 1$, and for every $x_0^n \neq x \in \{0,1\}^n$, $C_{x_0^n}(x) = 0$. Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}} \cup \{C_{x_0^n}\}_{n \in \mathbb{N}, x_0^n \in \{0,1\}^n}$, and assume that for every $n \in \mathbb{N}$ and $x_0^n \in \{0,1\}^n$, $|C_n| = \left|C_{x_0^n}\right|$. (This is without loss of generality, since all circuits taking n-bit inputs can be padded to have the same size as the circuit with maximal size.) For every $n \in \mathbb{N}$, let $f_n(\{k_i\}) = \sum_{x \in \{0,1\}^n} E(\{k_i\}_i, x)$, and $\mathcal{F} = \{f_n\}$, then $\mathcal{F} \in \mathrm{VNP}$ by definition. We show that $\mathcal{F} \notin \mathrm{VP}$, by showing that a PPT simulator cannot simulate zero-test queries to $f_n(\{k_i\})$, where $\{k_i\}$ are the encodings in an obfuscation of a circuit from the family $\mathcal{C}$. Indeed, if such zero-test queries cannot be simulated then the VBB property of $\mathcal{O}$ guarantees that no polynomial-time adversary can construct circuits computing the functions f_n, namely $\mathcal{F}$ has no polynomial-sized circuits.

For a PPT adversary $\mathcal{A}$, let Sim be the PPT simulator whose existence is guaranteed by the VBB property of $\mathcal{O}$, and let $t(n) : \mathbb{N} \to \mathbb{N}$ be the polynomial bounding the running time of Sim. In

particular, when simulating the obfuscation of any $C_{x_0^n}$, Sim makes at most $t(n)$ queries to its oracle. Let $x_0^n \in_R \{0,1\}^n$, then the view of Sim when given $1^{|C_{x_0^n}|}$, and oracle access to $C_{x_0^n}$, is identical to its view when given $1^{|C_n|}$, and oracle access to C_n, *unless in any of these simulations,* Sim *queries its oracle on* x_0^n. As this happens with at most $\frac{t(n)}{2^n} = \mathsf{negl}(n)$ probability, then except with negligible probability Sim would return the same answer to the zero-test of f_n on the obfuscation of $C_{x_0^n}, C_n$, which would be wrong for one of them, in contradiction to the VBB property of $\mathcal{O}$. Therefore, $\mathcal{A}$ cannot construct a circuit computing f_n. □

4.1 The p-Bounded Speedup Hypothesis

We first state our new hypothesis. which is a worst-case assumption that is inspired by the Bounded Speedup Hypothesis (BSH) of [BR14a, BR14b]. More specifically, it corresponds exactly to replacing 3SAT with Max-2-SAT in the BSH, 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 4.4 (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 $\geq 7/10$ fraction of ϕ's clauses.

Assumption 4.5 (p-Bounded Speedup Hypothesis)**.** Let $p : \mathbb{N} \to \mathbb{N}$. Then for any X-Max-2-SAT solver that has size $t(n)$, $|X| \leq p(\mathsf{poly}(t(n)))$.

Intuitively, the strongest form of Assumption 4.5, namely when $p = \mathsf{poly}(n)$, states that MAX-2-SAT is exponentially hard even on $2^{\Omega(n)}$-size subsets of $\{0,1\}^n$. The BSH [BR14a, BR14b] states the same for 3SAT, thus strengthening the well-studied Exponential Time Hypothesis (ETH) introduced by Impagliazzo and Paturi [IP99]. Recall that the ETH states that there exists no $2^{o(n)}$-time algorithm for deciding 3SAT over $\{0,1\}^n$, which also implies the same for many other NP complete problems.

Subsequent to the publication of [BR14a, BR14b], the BSH was shown to be false by Uri Feige using a SAT-solver based attack [Rot15]. We note that this attack does not directly apply to Assumption 4.5, which uses a different NP-complete problem. (In fact, our construction can be made to work with other NP-complete problems as well.) Still, in light of the attack, we choose to parameterize our assumption, such that even if its strongest form is false, weaker forms (e.g., when $p = 2^{\mathsf{poly}\log(n)}$) still give a meaningful result.

The proof of the following lemma is inspired by [BR14b, Lemma 3.14]).

Lemma 4.6. *Assume the p-Bounded Speedup Hypothesis. Then for any circuit e that computes a graded-multilinear polynomial, $m_e \leq p(\mathsf{poly}(|e|))$.*

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 $\mathsf{poly}(|e|)$. □

4.2 The zero-test simulator

At a high level, the strategy for simulating zero-test queries is the same as in the previous section (Construction 3.13). 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 4.7 gives an algorithm that extracts the list of full α-monomials; a similar algorithm was used in the zero-testing procedure of [BR14b].

Lemma 4.7. *Let e be a circuit computing a graded-multilinear 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 $\widetilde{\alpha}$. □

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 3.13. 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 $p = \mathsf{poly}(n)$ in the p-Bounded Speedup Hypothesis gives a $\mathsf{poly}(|e|)$-time VBB simulator). We omit further details, and this completes the proofs of Theorems 1.8 and 1.9.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 528–556, 2015.

[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.

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, pages 162–172, 2014.

[Bab87] László Babai. Random oracles separate PSPACE from the polynomial-time hierarchy. *Inf. Process. Lett.*, 26(1):51–53, 1987.

[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.

[BGH+15] Zvika Brakerski, Craig Gentry, Shai Halevi, Tancrède Lepoint, Amit Sahai, and Mehdi Tibouchi. Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845, 2015. http://eprint.iacr.org/.

[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.

[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/.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 247–266, 2015.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 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.

[FLMS14] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 128–135, 2014.

[FMM15] Hervé Fournier, Guillaume Malod, and Stefan Mengel. Monomials in arithmetic circuits: Complete problems in the counting hierarchy. *Computational Complexity*, 24(1):1–30, 2015.

[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.

[GGH14] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graded multilinear maps from lattices. Cryptology ePrint Archive, Report 2014/645, 2014.

[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.

[Hal15] Shai Halevi. Graded encoding, variations on a scheme. Cryptology ePrint Archive, Report 2015/866, 2015. http://eprint.iacr.org/.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. *IACR Cryptology ePrint Archive*, 2015:301, 2015.

[IP99] Russell Impagliazzo and Ramamohan Paturi. Complexity of k-SAT. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity, Atlanta, Georgia, USA, May 4-6, 1999*, pages 237–240, 1999.

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

[NW97] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997.

[Rot15] Guy Rothblum. Personal communication, 2015.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484, 2014.

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

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 439–467, 2015.

A Proof of Lemma 3.15

In this section we prove Lemma 3.15.

Proof of Lemma 3.15. 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 3.16), 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,

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

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

□