Obfuscation without the Vulnerabilities of Multilinear Maps*

Sanjam Garg† Pratyay Mukherjee‡ Akshayaram Srinivasan§

April 19, 2016

Abstract

Indistinguishability obfuscation is a central primitive in cryptography. Security of existing multilinear maps constructions on which current obfuscation candidates are based is poorly understood. In a few words, multilinear maps allow for checking if an arbitrary bounded degree polynomial on hidden values evaluates to zero or not. All known attacks on multilinear maps depend on the information revealed on computations that result in encodings of zero. This includes the recent annihilation attacks of Miles, Sahai and Zhandry [EPRINT 2016/147] on obfuscation candidates as a special case.

Building on a modification of the Garg, Gentry and Halevi [EUROCRYPT 2013] multilinear maps (GGH for short), we present a new obfuscation candidate that is resilient to these vulnerabilities. Specifically, in our construction the results of all computations yielding a zero *provably* hide all the secret system parameters. This is the first obfuscation candidate that weakens the security needed from the zero-test.

Formally, we prove security of our construction in a weakening of the idealized graded encoding model that accounts for *all known* vulnerabilities on GGH maps.

*Research supported in part from a DARPA/ARL SAFEWARE award, AFOSR Award FA9550-15-1-0274, and NSF CRII Award 1464397. 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.

†University of California, Berkeley, sanjamg@berkeley.edu

‡University of California, Berkeley, pratyay85@berkeley.edu

§University of California, Berkeley, akshayaram@berkeley.edu

1 Introduction

The goal of program obfuscation is to make computer programs "unintelligible" without affecting their functionality. Defining obfuscation formally turns out to be rather tricky. It was first formalized by Hada [Had00] and Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan and Yang [BGI+01]. The first *candidate* construction for obfuscation was provided by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH+13b]. Subsequently, several other candidates [BR14, BGK+14, PST14, AGIS14, Zim15, AB15, GLSW15, Lin16] have been proposed. However, all these constructions rely on the conjectured security of new computational assumptions on multilinear maps [GGH13a, CLT13, GGH15], the security of which is poorly understood.

In short, multilinear maps allow homomorphic computation of a priori bounded degree (say κ) arbitrary polynomials on "encrypted" values (in $R/\mathcal{I}$ in the case of [GGH13a], where $R = \mathbb{Z}[X]/(X^n+1)$ and $\mathcal{I}$ is a prime ideal in R) and a mechanism for publicly checking if the polynomial yields a zero or not. From a high-level, obfuscation of a program involves mapping the program to a sequence of values in $R/\mathcal{I}$ which are then encrypted using multilinear maps. Evaluation of the program on a specific input y corresponds to evaluating a public polynomial p_y (of degree κ) on the encrypted values and checking if it yields a zero or not.

Starting with the multilinear map candidate of Garg, Gentry and Halevi [GGH13a], two other proposals [CLT13, GGH15] have been made and a few variants of them [LSS14, CLT15] are also considered. Several vulnerabilities in all these candidate constructions have been demonstrated [GGH13a, CHL+15, HJ15, CGH+15, CLLT15, MSZ16]. We are most interested in the attacks against the GGH construction [GGH13a, HJ15, MSZ16]. One common aspect of all these three attacks on GGH (and in principle, also for the attacks against the other multilinear map candidates) is that they arise when the attacker gets hold of the so-called "zero-encodings" or encryptions of zero in the above description.

These attacks on GGH can be categorized into two broad classes. The basic ones [GGH13a, HJ15] where "low-level"[1] zero-encodings are provided (or can be obtained) and the more sophisticated ones [MSZ16] where zero-encodings can be obtained only at the "highest-level." Nonetheless, both categories of attacks follow the same generic strategy of applying the zero-testing procedure on the zero-encodings that reveals ring elements (in the ring R). Known attacks exploit the correlations among these ring elements. These correlations are much harder to leverage in the case where only "highest-level" zero-encodings can be obtained, which is the case for known obfuscation candidates. The only known attacks against obfuscation schemes are the recent annihilation attacks of Miles, Sahai and Zhandry [MSZ16]. However, not all the obfuscation candidates are broken by the annihilation attacks (See [AJN+16, Appendix A] for a detailed account on the current status of the iO constructions). For example, it is not known if the annihilation attacks directly extend to the first iO candidate of Garg et al. [GGH+13b]. The difficulty stems from the extra "defenses" they apply, namely appending random elements to the diagonal of the branching program. This randomization of the branching program makes it hard to find an annihilating polynomial. However, since this (and other obfuscation candidates that are not immediately broken) weren't constructed to handle this direction of attacks, it will not be surprising if a more sophisticated version of annihilation breaks these obfuscation candidates as well in near future. This raises the following natural question:

[1] Here "low-level" encodings are such that more computation can be performed on it whereas no further computation can be performed on an encoding at "highest-level."

Can we realize an obfuscation candidate that provably defends against attacks that exploit information leaked from any zero-encoding?

Our Result. In this work, we provide a new obfuscation candidate that resists such attacks. Our construction is based on a modification of the GGH multilinear map, which is crucial for the construction and its security. We argue security in the so-called *hybrid graded encoding model* that captures *all known attacks* against the GGH map.[2]

1.1 Technical Ideas: Self-Fortifying Obfuscation

GGH Multilinear Maps. We start by recalling the GGH multilinear maps briefly. An instance of the GGH scheme is parameterized by the security parameter λ and the required multi-linearity level $\kappa \leq \text{poly}(\lambda)$. Based on these parameters, consider the $2n$-th cyclotomic ring $R = \mathbb{Z}[X]/(X^n+1)$ where n is a power of 2 (n is set to be "large enough" to ensure security), and a modulus q that defines $R_q = R/qR$ (with q large enough to support functionality). An instance of the GGH scheme relative to the parameters above encodes elements of a quotient ring $R/\mathcal{I}$, where $\mathcal{I}$ is a principal prime ideal $\mathcal{I} = \langle \boldsymbol{g} \rangle \subset R$, generated by a "short" vector $\boldsymbol{g}$. Namely, the "ring elements" that are being encoded are cosets of the form $\boldsymbol{e} + \mathcal{I}$ for some vector $\boldsymbol{e}$. The short generator $\boldsymbol{g}$ itself is kept secret, and no "good" description of $\mathcal{I}$ is made public in this scheme. In addition, GGH system depends on another secret element $\boldsymbol{z}$, which is chosen at random in R_q (and hence is not short).

A level-zero ("plaintext") encoding of a coset $\boldsymbol{e} + \mathcal{I} \in R/\mathcal{I}$ is just a short vector in that coset. For higher-level encodings, a level-i encoding of the same coset is a vector of the form $[\boldsymbol{c}/\boldsymbol{z}]_q$ (We use the notation $[\cdot]_q$ to denote operations in R_q.) with $\boldsymbol{c} \in \boldsymbol{e} + \mathcal{I}$ short. The GGH encoding scheme provides a public zero-testing parameter to check if any given level-κ encoding is an encoding of 0. This zero-testing parameter is of the form $\boldsymbol{p}_{\mathsf{zt}} = [\boldsymbol{h}\boldsymbol{z}^\kappa/\boldsymbol{g}]_q$ where $\boldsymbol{h}$ is of size $\sqrt{q}$. System parameters are setup in a way so that $\boldsymbol{u}$ is an encoding of zero if and only if $\boldsymbol{u} \cdot \boldsymbol{p}_{\mathsf{zt}}$ is "small."

Template of attacks. As mentioned earlier, all known attacks [GGH13a, HJ15, MSZ16] on the GGH map exploit the correlations among the ring elements obtained after performing zero-tests on several encodings of 0. Among the attacks, the most potent ones against the candidate obfuscations are the annihilation attacks [MSZ16]. For certain obfuscation candidates, this attack can distinguish between obfuscations of two specially constructed functionally equivalent branching programs, say BP_0 and BP_1. We now give an intuitive explanation of the attack.

An attacker against the obfuscation candidate proceeds by evaluating the challenge obfuscation (which is either an obfuscation of BP_0 or BP_1) on specific inputs $x_1, x_2, \cdots$ (where the branching program evaluates to 0) to obtain "top-level" encodings of zero. Any such encoding of zero corresponding to the input x_i is of the form:

$$e_i = \frac{\boldsymbol{\gamma_{i,1}} \cdot \boldsymbol{g} + \boldsymbol{\gamma_{i,2}} \cdot \boldsymbol{g}^2 + \cdots + \boldsymbol{\gamma_{i,\kappa}} \cdot \boldsymbol{g}^\kappa}{\boldsymbol{z}^\kappa}$$

where $\boldsymbol{\gamma_{i,j}}$'s are ring elements that depend on the obfuscated program as well as the randomness generated by the encoding procedure of the GGH multilinear maps. The attacker then zero-tests

[2] Our model is a *strict weakening* of the ideal graded encoding model considered in the previous works [BGK+14, AGIS14, BMSZ16]. We note that the ideal graded encoding model, as the name suggests, assumes no vulnerability in the underlying multi-linear map construction.

on the encoding e_i to obtain a ring element $\boldsymbol{f}_i$ given by:

$$\boldsymbol{f}_i = [e_i \cdot \mathsf{p_{zt}}]_q = \boldsymbol{h} \cdot \left(\boldsymbol{\gamma_{i,1}} + \boldsymbol{\gamma_{i,2}} \cdot \boldsymbol{g} + \cdots + \boldsymbol{\gamma_{i,\kappa}} \cdot \boldsymbol{g}^{\kappa-1}\right)$$

The main crux of the attack is to come up with a specific polynomial p (called as the *annihilating polynomial*) such that $p(\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots)$ is *identically* 0 modulo $\mathcal{I}$ for BP_0 and is *not identically* 0 for BP_1 which obviously enables the attacker to distinguish between obfuscations of BP_0 and BP_1.

Our Idea. Our high level idea for avoiding this attack is to set up the obfuscation in a manner such that the ring elements obtained via such successful zero-tests are "uncorrelated", at least against a computationally bounded attacker.

In more detail, in our new construction, any "top-level" encoding of zero (corresponding to an honest evaluation of the obfuscation on an input x_i) is of the form:

$$e'_i = \frac{(\boldsymbol{\gamma_{i,1}} + \boldsymbol{\beta_i}) \cdot \boldsymbol{g} + \boldsymbol{\gamma_{i,2}} \cdot \boldsymbol{g}^2 + \cdots + \boldsymbol{\gamma_{i,\kappa}} \cdot \boldsymbol{g}^{\kappa}}{\boldsymbol{z}^{\kappa}} \tag{1.1}$$

where $\boldsymbol{\beta}_i$ is a "blinding" factor. This is essentially output of a special PRF computation[3] (on the same input x_i) that occurs within our obfuscation construction. As a result, a successful zero-test like above on e'_i would yield a value $\boldsymbol{f}'_i = \boldsymbol{\gamma_{i,1}} + \boldsymbol{\beta_i} \bmod \mathcal{I}$ which is a computationally close to a uniform random value $\bmod \mathcal{I}$. For several such inputs $x_1, x_2, \ldots$ the adversary obtains $\boldsymbol{f}'_1, \boldsymbol{f}'_2, \ldots$ all of them being close to uniform random $\bmod \mathcal{I}$ and hence uncorrelated to each other to a computationally bounded adversary. Notice that, here we invoke a new strategy, that we call "self-fortification", to modify the obfuscation construction such that it evaluates a pseudorandom function within itself in addition to the main computation. Realizing this intuition involves building a new obfuscation construction and a corresponding multilinear maps that support it.

Modifying GGH Maps and realizing our construction. Recall that the GGH construction supports computation only in the ring $R/\mathcal{I}$ where $\mathcal{I} = \langle \boldsymbol{g} \rangle$. Therefore, the structure of computation outside this ring is not preserved. In our construction, we want to randomize the ring elements that are obtained after a zero-test. This corresponds to computing a PRF value in the ideal of $\boldsymbol{g}$ i.e we want the obfuscation to compute $\boldsymbol{g} \cdot \mathsf{PRF}(x)$ on an input x alongside the computation of the actual program. Therefore, we modify the GGH construction to support this change. In particular, we set the multilinear maps in such a way that all encodings are generated in a larger ring $R/\mathcal{J}$ where $\mathcal{J} = \langle \boldsymbol{g}^2 \rangle$. On the other hand, identical to GGH, zero-test is done in the smaller ring $R/\mathcal{I}$. Concretely, our zero-testing procedure checks if the final encoding is $0 \bmod \mathcal{I}$.

Notice that due to the special choice of the ideals even if the encodings are generated in $R/\mathcal{J}$, the main computation (corresponding to the given program) is preserved both in $R/\mathcal{J}$ as well as in $R/\mathcal{I}$. On the other hand, the pseudorandom function computation is preserved only in $R/\mathcal{J}$ but not in $R/\mathcal{I}$ as it is computed only in those cosets of $\mathcal{J}$ which are in the ideal of $\mathcal{I}$. Therefore even if we generate encodings in $R/\mathcal{J}$ our zero-test works only in $R/\mathcal{I}$. In addition to that, when zero-test succeeds, the revealed ring element is "masked" by the output of the pseudorandom function. Hence, as observed above, the adversary ends up obtaining uniform random values (in $R/\mathcal{I}$) corresponding to different inputs rendering it infeasible to launch an annihilation attack.

[3] Looking ahead, the PRF computation outputs a pseudorandom value modulo $\mathcal{I}$.

One additional technical difficulty shows up in the choice of the pseudorandom function. Looking ahead, to prove security in our model, we need the output of the pseudorandom function to be random $\mathsf{mod}\,\mathcal{I}$. A natural method for that would be to use Gaussian sampling. However, the length of the branching program for performing this sampling, grows with n. This forces the multilinearity needed to grow with n. This ultimately ends up creating a circularity in how the parameters q and n need to grow in order to ensure functionality and preserve security respectively. Instead we present an alternative (and much simpler) sampling procedure for which the length of the branching program is independent of n. This allows to avoid this circularity issue. We refer the reader to the main body of the paper for details.

1.2 Our Model: Hybrid Graded Encoding

We prove that our construction is (VBB) secure in the hybrid[4] graded encoding model, that is a weakening of the ideal graded encoding model used in the previous works [BGK+14, BMSZ16, AGIS14]. We remark that the major difference is that the ideal graded encoding model assumes no vulnerability in the underlying multi-linear map construction whereas our model captures *all known vulnerabilities* found till date to the best of our knowledge against the GGH encoding.

A brief overview. First we provide an informal overview of our model and then briefly explain how it captures the existing attacks.

In order to capture all known attacks on the GGH encoding scheme we explicitly work over the $2n$-th cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$ and the ideal $\mathcal{I} = \langle \boldsymbol{g} \rangle$ where $\boldsymbol{g}$ is a "short" element in R. Similar to [BGK+14], our encodings are leveled with respect to the sets $\subseteq \mathbb{U}$ for some universe $\mathbb{U}$. An encoding e is denoted as $[\boldsymbol{t}]_S$ for some $\boldsymbol{t} \in R$ and $S \subseteq \mathbb{U}$. Looking ahead, $\boldsymbol{t}$ is a "short ring element" representing the coset to be encoded. For any such e the element $\boldsymbol{t}$ is called its *representation* and S its *level*.

Similar to the ideal model, we implement our security model via an oracle which does not give the adversary direct access to any encoding, instead only gives a corresponding handle. It maintains a handle-table where it stores all the encoding-handle pairs generated at any time. The adversary can compute arbitrary algebraic operations querying via the handles. Like the ideal model, the adversary is allowed to zero-test. However, while in the ideal model the oracle just returns whether the zero-test succeeds or not, here we go one step further. If any zero-test succeeds our oracle returns a second type of handle, called ring-handle corresponding to the ring element resulting from that successful zero-test. Moreover, we allow the adversary to perform arbitrary bounded-degree polynomial computations on those ring elements via the corresponding ring-handles. Formalizing this a bit more, the adversary can make the following queries via handles.

- *Addition* in the same level as $[\boldsymbol{t}_1]_S + [\boldsymbol{t}_2]_S \to [\boldsymbol{t}_1 + \boldsymbol{t}_2]_S$. The oracle returns a new handle corresponding to $[\boldsymbol{t}_1 + \boldsymbol{t}_2]_S$
- *Multiplication* among the disjoint levels as $[\boldsymbol{t}_1]_{S_1} \cdot [\boldsymbol{t}_2]_{S_2} \to [\boldsymbol{t}_1 \cdot \boldsymbol{t}_2]_S$ (where $S_1 \cap S_2 = \emptyset$ and $S = S_1 \cup S_2$). The oracle returns a new handle corresponding to $[\boldsymbol{t}_1 \cdot \boldsymbol{t}_2]_{S_1 \cup S_2}$.

[4]Note that our model lies somewhere in between the actual GGH encoding and the ideal graded encoding and hence called *hybrid*.

- *Ring multiplication*, that is multiplication of an encoding $[\boldsymbol{t}]_S$ with an arbitrary ring element $\boldsymbol{r} \in R$ to produce an encoding $[\boldsymbol{t} \cdot \boldsymbol{r}]_S$, corresponding to which the oracle returns a new handle.[5]

- *Zero-testing* an encoding $[\boldsymbol{t}]_S$ which succeeds if and only if $\boldsymbol{t} = 0 \pmod{\mathcal{I}}$ and $S = \mathbb{U}$. Now only if it succeeds the oracle returns a ring-handle to the element $\boldsymbol{a}$ where $\boldsymbol{t} = \boldsymbol{ag}$ (this form is guaranteed by the zero-test).

- *Post-zeroizing computations* on the ring elements $\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots$ all of which are produced via successful zero-tests. The adversary simply asks the oracle to compute a non-zero bounded-degree polynomial over those elements via corresponding ring-handles. If the evaluation outputs an element in the ideal $\mathcal{I}$ then the adversary wins.[6]

We remark that if in any case adversary queries on any handle that is not found in the corresponding handle-table then the oracle returns nothing which ensures that all the handles were already generated by the oracle before.

Capturing existing attacks. Now we briefly discuss how our model captures all existing attacks against the GGH multilinear map. First note that, an encoding of an element $\boldsymbol{r} \in R$ in the GGH construction can be expressed as $[\boldsymbol{t}]_S$ such that $\boldsymbol{t} = \boldsymbol{r} \pmod{\mathcal{I}}$. It is easily observed that any attack which uses low-level encodings[7] of zero (referred to as the "basic attacks" earlier) immediately extends to our model as follows: given handles for two zero-encodings $e_1 = [\boldsymbol{a}_1\boldsymbol{g}]_{S_1}$ and $e_2 = [\boldsymbol{a}_2\boldsymbol{g}]_{S_2}$ for some $\boldsymbol{a}_1, \boldsymbol{a}_2 \in R$ and $S_1 \cap S_2 = \emptyset$ and another arbitrary encoding $e' = [\boldsymbol{u}]_{\mathbb{U}\setminus\{S_1 \cup S_2\}}$ for some $\boldsymbol{u} \in R \setminus \mathcal{I}$ the adversary first asks the oracle to compute the product $\widetilde{e} = e_1 \cdot e_2 \cdot e' = \left[\boldsymbol{cg}^2\right]_{\mathbb{U}}$ for some $\boldsymbol{c} \in R$. Subsequently, zero-testing on $\widetilde{e}$ returns a ring-handle to the element $\boldsymbol{cg} \in \mathcal{I}$ that belongs to the ideal. Therefore adversary wins trivially by providing the ring-handle even without any non-trivial computation on that. Recall that both the weak discrete log attack [GGH13a] and the attack by Hu and Jia [HJ15] on the multi-linear map based key-agreement scheme [GGH13a] inherently requires low-level zeros and hence can be easily launched successfully in our model.

Furthermore, the most recent (and seemingly more powerful) attack, namely the annihilation attack [MSZ16], can also be successfully executed in our model. The annihilation attack is applicable to several candidate obfuscation schemes [BGK+14, BMSZ16, AGIS14] that are based on the GGH map. In those constructions, the obfuscator just outputs the GGH encodings of all the elements in the branching program.[8] Recall that, the main idea behind an annihilation attack is to first evaluate a program on several inputs, all of which leads to encodings of zeros in the top-level, and then compute a so-called annihilation polynomial on the corresponding elements produced as a result of successful zero-tests on those top-level zeros. For specific programs the annihilation polynomial evaluates to an element in the ideal.

[5] Notice that in the ideal graded encoding model such operation is implicit as they consider integer rings and hence an integer multiplication is equivalent to multiple additions.

[6] In order to formalize the winning event of the adversary, we make the oracle output its entire state including the secret $\boldsymbol{g}$ and the actual encodings in this case. For any meaningful security definition such event must trigger a total break as it is impossible to simulate these values.

[7] Such encodings are provided publicly for a public encoding procedure which is necessary for certain applications of multi-linear maps. However, the obfuscation constructions do not require such feature. Hence those basic attacks do not extend to the obfuscation setting.

[8] In reality the branching programs are first randomized and then encoded, but here we ignore the randomizers for simplicity of exposition.

Now we briefly explain how such attack can be launched against those obfuscation constructions in our hybrid model. Consider the branching program consisting only of identity matrices, a specific set of inputs $\mu_1, \mu_2, \ldots, \mu_\tau$ identical to the inputs considered by [MSZ16] and the corresponding annihilation polynomial p. Clearly on those inputs (in fact on any input) the branching program trivially evaluates to 0. Now initially the adversary gets hold on all the encodings generated by the obfuscator via encoding handles. Observe that, a correct evaluation of the encoded program on any input μ_i leads to an encoding of zero, say $e_i = [\boldsymbol{a}_i \boldsymbol{g}]_{\mathbb{U}}$ at the top level for some $\boldsymbol{a}_i \in R$. Such evaluation can be performed through the algebraic queries with the corresponding encoding handles. Subsequently, a zero-test query on such e_i returns a ring-handle to $\boldsymbol{a}_i$. Once the adversary completes the zero-test queries for all inputs $\mu_1, \ldots \mu_\tau$ it gets the ring-handles corresponding to $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_\tau$. Finally the adversary queries a post-zeroizing computation asking the oracle to compute the annihilation polynomial p over the ring elements $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_\tau$ via the ring-handles. As shown by [MSZ16], such annihilation polynomial outputs a non-trivial element in the ideal $\mathcal{I}$ and hence the adversary wins in our model.

Proving security. Now we briefly explain the main ideas in the security argument. Recall that, our modified encoding procedure encodes in $R/\langle \boldsymbol{g}^2 \rangle$. Therefore, in our hybrid model an encoding $[\boldsymbol{t}]_S$ is corresponding to a coset $\boldsymbol{t} \pmod{\boldsymbol{g}^2}$. Also recall that, our construction consists of two parts. The main part evaluates the main branching program (on an input) that we want to obfuscate. Additionally we put a auxiliary branching program which evaluates a PRF (on the same input). The PRF computation outputs a "random" element modulo the ideal $\langle \boldsymbol{g} \rangle$ whereas the main program outputs a ring element.[9] Now, we encode the combined branching program using the modified GGH encoding.[10] Performing the computation correctly on some input x yields an encoding $e_x = [\boldsymbol{w}_{\mathsf{main}} + \boldsymbol{g}\boldsymbol{w}_{\mathsf{aux}}]_{\mathbb{U}}$ where $\boldsymbol{w}_{\mathsf{main}}$ is the output of the main program and $\boldsymbol{w}_{\mathsf{aux}}$ is the output of the PRF computation both on input x. The pseudorandomness of the PRF ensures that the value $\boldsymbol{r} := \boldsymbol{w}_{\mathsf{aux}} \pmod{\boldsymbol{g}}$ is computationally close to a uniform random sample element in $R/\langle \boldsymbol{g} \rangle$. So, in our model a zero-test query on e_x would make the oracle return a ring-handle to the ring element $\boldsymbol{r} + \boldsymbol{c}\boldsymbol{g}$ such that $\boldsymbol{r}$ is computationally indistinguishable from a uniform random sample from $R/\langle \boldsymbol{g} \rangle$. Therefore, repeating the above steps many times for different inputs $x_1, x_2, \ldots$ the adversary only gets ring-handles to the ring elements $\boldsymbol{r}_1 + \boldsymbol{c}_1\boldsymbol{g}, \boldsymbol{r}_2 + \boldsymbol{c}_2\boldsymbol{g}, \cdots$ each of which is computationally indistinguishable from a uniform random element in the ring $R/\langle \boldsymbol{g} \rangle$ and have no correlations (except with negligible probability) to a computationally bounded attacker. Now since the post-zeroizing computation is done in $R/\langle \boldsymbol{g} \rangle$, intuitively we can conclude that the no post-zeroizing computation outputs an element in the ideal $\mathcal{I}$ except with a negligible probability using Schwartz-Zippel lemma.

We remark that, however, the formal proof is much involved as we have to argue that the adversary can not learn anything "useful" by any other means, for example by some partial/incorrect computations. Nonetheless, formalizing above intuition and combining that with the formal techniques from the previous works we are able to prove that our construction is indeed VBB secure in the hybrid graded encoding model. The formal proof is presented in Section 8.

[9] We generalize the notion of Branching programs to work over arbitrary rings instead of $\{0, 1\}$.

[10] We ignore the Killian randomization here for simplicity of our exposition.

2 Notations

The natural security parameter throughout this paper is λ, and all other quantities are implicitly assumed to be functions of λ. We use standard big-O notation to classify the growth of functions, and say that $f(\lambda) = \tilde{O}(g(\lambda))$ if $f(\lambda) = O(g(\lambda) \cdot log^c\ \lambda)$ for some fixed constant c. We let $\mathsf{poly}(\lambda)$ denote an unspecified function $f(\lambda) = O(\lambda^c)$ for some constant c. A *negligible* function, denoted generically by $\mathsf{negl}(\lambda)$, is an $f(\lambda)$ such that $f(\lambda) = o(\lambda^{-c})$ for every fixed constant c. We say that a function is *overwhelming* if it is $1 - \mathsf{negl}(\lambda)$.

The *statistical distance* between two distributions X and Y over a domain D is defined to be $\frac{1}{2}\sum_{d\in D} |\Pr[X = d] - \Pr[Y = d]|$. We say that two ensembles of distributions $\{X_\lambda\}$ and $\{Y_\lambda\}$ are *statistically indistinguishable* if for every λ the statistical distance between X_λ and Y_λ is negligible in λ.

Two ensembles of distributions $\{X_\lambda\}$ and $\{Y_\lambda\}$ are *computationally indistinguishable* if for every probabilistic poly-time non-uniform (in λ) machine $\mathcal{A}$, $|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]|$ is negligible in λ. The definition is extended to non-uniform families of poly-sized circuits in the standard way.

3 Preliminaries for our modified GGH construction

Most parts of this section are taken verbatim from [Gar15]. We keep this part for completeness.

3.1 Lattices

We denote set of complex number by $\mathbb{C}$, real numbers by $\mathbb{R}$, the rationals by $\mathbb{Q}$ and the integers by $\mathbb{Z}$. For a positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. By convention, vectors are assumed to be in column form and are written using bold lower-case letters, e.g. $\boldsymbol{x}$. The ith component of $\boldsymbol{x}$ will be denoted by x_i. We will use $\boldsymbol{x}^T$ to denotes the transpose of $\boldsymbol{x}$. For a vector $\boldsymbol{x}$ in $\mathbb{R}^n$ or $\mathbb{C}^n$ and $p \in [1, \infty]$, we define the ℓ_p norm as $\|\boldsymbol{x}\|_p = \left(\sum_{i\in[n]} |x_i|^p\right)^{1/p}$ where $p < \infty$, and $\|\boldsymbol{x}\|_\infty = \max_{i\in[n]} |x_i|$ where $p = \infty$. Whenever p is not specified, $\|\boldsymbol{x}\|$ is assumed to represent the ℓ_2 norm (also referred to as the Euclidean norm).

Matrices are written as bold capital letters, e.g. $\boldsymbol{X}$, and the ith column vector of a matrix $\boldsymbol{X}$ is denoted $\boldsymbol{x}_i$. Finally we will denote the transpose and the inverse (if it exists) of a matrix $\boldsymbol{X}$ with $\boldsymbol{X}^T$ and $\boldsymbol{X}^{-1}$ respectively.

A lattice Λ is an additive discrete sub-group of $\mathbb{R}^n$, i.e., it is a subset $\Lambda \subset \mathbb{R}^n$ satisfying the following properties:

(subgroup) Λ is closed under addition and subtraction,

(discrete) there is a real $\varepsilon > 0$ such that any two distinct lattice points $\boldsymbol{x} \neq \boldsymbol{y} \in \Lambda$ are at distance at least $\|\boldsymbol{x} - \boldsymbol{y}\| \geq \varepsilon$.

Let $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\} \subset \mathbb{R}^n$ consist of k linearly independent vectors in $\mathbb{R}^n$. The lattice generated by the $\boldsymbol{B}$ is the set

$$\mathcal{L}(\boldsymbol{B}) = \{\boldsymbol{Bz} = \sum_{i=1}^{k} z_i \boldsymbol{b}_i : \boldsymbol{z} \in \mathbb{Z}^k\},$$

of all the integer linear combinations of the columns of $\boldsymbol{B}$. The matrix $\boldsymbol{B}$ is called a *basis* for the lattice $\mathcal{L}(\boldsymbol{B})$. The integers n and k are called the *dimension* and *rank* of the lattice. If $n = k$ then $\mathcal{L}(\boldsymbol{B})$ is called a *full-rank* lattice. We will only be concerned with full-rank lattices, hence unless otherwise mentioned we will assume that the lattice considered is full-rank.

For lattices $\Lambda' \subseteq \Lambda$, the quotient group Λ/Λ' (also written as $\Lambda \bmod \Lambda'$) is well-defined as the additive group of distinct *cosets* $\boldsymbol{v} + \Lambda'$ for $\boldsymbol{v} \in \Lambda$, with addition of cosets defined in the usual way.

3.2 Gaussians on Lattices

Review of Gaussian measure over lattices presented here follows the development by prior works [Reg04, AR05, MR07, GPV08, AGHS12]. For any real $s > 0$, define the (spherical) *Gaussian function* $\rho_s : \mathbb{R}^n \to (0, 1]$ with[11] parameter s as:

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \rho_s(\boldsymbol{x}) = \exp(-\pi\langle \boldsymbol{x}, \boldsymbol{x}\rangle/s^2) = \exp(-\pi\|\boldsymbol{x}\|^2/s^2).$$

For any real $s > 0$, and n-dimensional lattice Λ, define the (spherical) *discrete Gaussian distribution* over Λ as:

$$\forall \boldsymbol{x} \in \Lambda, D_{\Lambda,s}(\boldsymbol{x}) = \frac{\rho_s(\boldsymbol{x})}{\rho_s(\Lambda)}.$$

Smoothing Parameter. Micciancio and Regev [MR07] introduced a lattice quantity called the *smoothing parameter*, and related it other lattice parameters.

Definition 3.1 (Smoothing Parameter, [MR07, Definition 3.1]) *For an n-dimensional lattice Λ, and positive real $\varepsilon > 0$, we define its* smoothing parameter *denoted $\eta_\varepsilon(\Lambda)$, to be the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \leq \varepsilon$.*

Intuitively, for a small enough ε, the number $\eta_\varepsilon(\Lambda)$ is sufficiently larger than a fundamental parallelepiped of Λ so that sampling from the corresponding Gaussian "wipes out the internal structure" of Λ. The following Lemma 3.2 formally provide this claim. Finally Lemma 3.3 provides bounds on the length of a vector sampled from a Gaussian.

Lemma 3.2 ([GPV08, Corollary 2.8]) *Let Λ, Λ' be n-dimensional lattices, with $\Lambda' \subseteq \Lambda$. Then for any $\varepsilon \in (0, \frac{1}{2})$, any $s \geq \eta_\varepsilon(\Lambda')$, the distribution of $(D_{\Lambda,s} \pmod{\Lambda'})$ is within a statistical distance at most 2ε of uniform over $(\Lambda \pmod{\Lambda'})$.*

Lemma 3.3 ([MR07, Lemma 4.4] and [BF11, Proposition 4.7]) *For any n-dimensional lattice Λ, and $s \geq \eta_\varepsilon(\Lambda)$ for some negligible ε, then for any constant $\delta > 0$ we have*

$$\Pr_{\boldsymbol{x} \leftarrow D_{\Lambda,s}}\left[(1-\delta)s\sqrt{\frac{n}{2\pi}} \leq \|\boldsymbol{x}\| \leq (1+\delta)s\sqrt{\frac{n}{2\pi}}\right] \geq 1 - \mathsf{negl}(n).$$

[11] The Gaussian function can be defined more generally as being centered around a specific vector $\boldsymbol{c}$ instead of $\mathbf{0}$ as done here. The simpler definition considered here suffices for our purposes.

3.3 Number Fields and Ring of Integers

A *number field* can be defined as field extension $K = \mathbb{Q}(\zeta)$ obtained by adjoining an abstract element ζ to the field of rationals, where ζ satisfies the relation $f(\zeta) = 0$ for some irreducible polynomial $f(X) \in \mathbb{Q}[X]$, which is a monic (a polynomial whose leading coefficient is 1) polynomial without loss of generality. The polynomial $f(X)$ is called the *minimal polynomial* of ζ, and the *degree* n of the number field is the degree of f. Because $f(\zeta) = 0$, the number field K can be seen as an n-dimensional vector space over $\mathbb{Q}$ with basis $\{1, \zeta, \ldots, \zeta^{n-1}\}$. Associating ζ with indeterminate X yields an isomorphism between K and $\mathbb{Q}[X]/f(X)$.

The ring of integers $\mathcal{O}_K$, of a number field K of degree n, is a free $\mathbb{Z}$-module of rank n, i.e., the set of all $\mathbb{Z}$-linear combinations of some *integral basis* $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\} \subset \mathcal{O}_K$. Such a set is called an *integral basis*, and it is also a $\mathbb{Q}$-basis for K.

The case of Cyclotomic Number Fields. Let $\zeta_m = e^{2\pi\sqrt{-1}/m} \in \mathbb{C}$ denote a *primitive* m-th root of unity. (Recall that an mth root of unity is said to be a *primitive* root if it is not a kth root for some $0 < k < m$.) The m-th *cyclotomic polynomial*, denote by $\Phi_m(X)$, is defined as the product

$$\Phi_m(X) = \prod_{k \in \mathbb{Z}_m^*} (X - \zeta_m^k).$$

Observe that the values ζ^k run over all the primitive m^{th} roots of unity in $\mathbb{C}$, thus $\Phi_m(X)$ has degree $n = \varphi(m)$, where $\varphi(m)$ denotes the *Euler's totient* or *phi function*. Recall that if m is a positive integer, then $\varphi(m)$ is the number of integers in the set $\{1, 2, \ldots, m\}$ that are relatively prime to m.

The cyclotomic polynomial $\Phi_m(X)$ may be computed by (exactly) dividing $X^n - 1$ by the cyclotomic polynomials of the proper divisors of n previously computed recursively (setting, $\Phi_1(X) = X - 1$) by the same method:

$$\Phi_m(X) = \frac{X^m - 1}{\prod_{\substack{d|m \\ d<m}} \Phi_d(X)}.$$

We will be most interested in the case when $m \geq 2$ is a power of 2 in which case $\Phi_m(X) = X^{m/2} + 1$. The mth *cyclotomic field* $\mathbb{Q}(\zeta_m)$ (with $m > 2$) is obtained by adjoining ζ_m to $\mathbb{Q}$. The ring of integers in $\mathbb{Q}(\zeta_m)$ is $\mathbb{Z}(\zeta_m)$. This ring $\mathbb{Z}(\zeta_m)$ is called the *cyclotomic ring*.

Coefficient Embedding. There is also a *coefficient embedding* $\tau : K \to \mathbb{Q}^n$. As mentioned earlier, since $f(\zeta) = 0$, there is an isomorphism between $\mathbb{Q}[X] \pmod{f(X)}$ and K given by $X \to \zeta$. So, K can be represented as a n-dimensional vector space over $\mathbb{Q}$ using the *power basis* $\{1, \zeta, \ldots, \zeta^{n-1}\}$, and τ maps an element of K to its associated coefficient vector. When identifying an element $a \in K$ as a coefficient vector, i.e., $\tau(a)$ we denote it as a boldface vector $\boldsymbol{a}$. Note that the addition of vectors is done component-wise, while the multiplication is done as polynomials modulo $f(X)$. We define the *coefficient norm* of a as the norm of the vector $\boldsymbol{a}$. Specifically, we define the ℓ_p coefficient norm of a, denoted as $\|a\|_p$ or $\|\boldsymbol{a}\|_p$ as $\left(\sum_{i \in [n]} a_i^p\right)^{\frac{1}{p}}$ for $p < \infty$, and as $\max_{i \in [n]} |a_i|$ for $p = \infty$. (As always we assume the ℓ_2 norm when p is omitted.) We will use the following lemma.

Lemma 3.4 *Let $K = \mathbb{Q}[X]/(X^n + 1)$, for any positive integer n. $\forall \boldsymbol{a}, \boldsymbol{b} \in K$ and $\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$ we have that*

$$\|\boldsymbol{c}\| \leq \sqrt{n} \cdot \|\boldsymbol{a}\| \cdot \|\boldsymbol{b}\|.$$

Definition 3.5 (Ideal) *An* (integral) ideal $\mathcal{I} \subseteq \mathcal{O}_K$ *is a nontrivial (i.e., nonempty and nonzero*[12]*) additive subgroup that is closed under multiplication by $\mathcal{O}_K$ – that is, $r \cdot g \in \mathcal{I}$ for any $r \in \mathcal{O}_K$ and $g \in \mathcal{I}$. A* fractional ideal $\mathcal{I} \subset K$ *is a set such that $d \cdot \mathcal{I}$ is an integral ideal for some $d \in \mathcal{O}_K$. The* inverse $\mathcal{I}^{-1}$ *of an ideal $\mathcal{I}$ is the set $\{a \in K : a \cdot \mathcal{I} \subseteq \mathcal{O}_K\}$.*

Definition 3.6 *An ideal $\mathcal{I}$ is* principal *if $\mathcal{I} = \langle g \rangle$ for $g \in \mathcal{O}_K$ – that is, if one generator suffices.*

Invertibility of ring elements. Let R denote the $2n^{th}$ cyclotomic ring and let R_q denote R/qR for a prime q. We note that R_q is also a ring and not all elements in it are invertible. Let $R_q^\times$ denote the set of elements in R_q that are invertible. We next provide a lemma of Stehlé and Steinfeld that points out that a (large enough) random element is R_q is also in $R_q^\times$ with large probability.

Lemma 3.7 ([SS11, Lemma 4.1]) *Let $n \geq 8$ be a power of 2 such that X^n+1 splits into n linear factors modulo $q \geq 5$. Let $\sigma \geq \sqrt{n \ln(2n(1 + 1/\delta))/\pi} \cdot q^{1/n}$, for an arbitrary $\delta \in (0, 1/2)$. Then*

$$\Pr_{f \leftarrow D_{Z^n,\sigma}}[f \pmod q \notin R_q^\times] \leq n(1/q + 2\delta).$$

4 Modified GGH Graded Encodings

In this section we describe a modified version of the GGH graded encoding scheme. This modification is essential for our new obfuscation scheme.

GGH construction: Short Introduction. An instance of the GGH scheme is parameterized by the security parameter λ and the required multi-linearity level $\kappa \leq \text{poly}(\lambda)$. Based on these parameters, consider the $2n$th cyclotomic ring $R = \mathbb{Z}[X]/(X^n + 1)$ where n is a power of 2 (n is set large enough to ensure security), and a modulus q that defines $R_q = R/qR$ (with q large enough to support functionality). An instance of the GGH scheme relative to the parameters above encodes elements of a quotient ring $QR = R/\mathcal{I}$, where $\mathcal{I}$ is a principal prime ideal $\mathcal{I} = \langle \boldsymbol{g} \rangle \subset R$, generated by a "short" vector $\boldsymbol{g}$. Namely, the "ring elements" that are encoded are cosets of the form $\boldsymbol{e} + \mathcal{I}$ for some vector $\boldsymbol{e}$. The short generator $\boldsymbol{g}$ itself is kept secret, and no "good" description of $\mathcal{I}$ is made public in this scheme. In addition, GGH system depends on another secret element $\boldsymbol{z}$, which is chosen at random in R_q (and hence is not short).

A level-zero ("plaintext") encoding of a coset $\boldsymbol{e} + \mathcal{I} \in R/\mathcal{I}$ is just a short vector in that coset. For higher-level encodings, a level-i encoding of the same coset is a vector of the form $[\boldsymbol{c}/\boldsymbol{z}^i]_q$ (We use the notation $[\cdot]_q$ to denote operations in R_q.) with $\boldsymbol{c} \in \boldsymbol{e} + \mathcal{I}$ short. GGH provide a public zero-testing to check if any given level-κ encoding is an encoding of 0 (which suffices for checking equality). This zero-testing parameter is of the form $\boldsymbol{p}_{zt} = [\boldsymbol{h}\boldsymbol{z}^\kappa/\boldsymbol{g}]_q$ where $\boldsymbol{h}$ is of size $\sqrt{q}$. System parameters are setup in a way so that $\boldsymbol{u}$ is an encoding of zero if and only if $\boldsymbol{u} \cdot \boldsymbol{p}_{zt}$ is small.

[12] Some texts also define the trivial set $\{0\}$ as an ideal, but in this work it is more convenient to exclude it.

Our modification at high level. Our scheme is very similar to the original GGH scheme except that unlike the GGH scheme which encodes elements of the quotient ring $R/\mathcal{I}$ we will encode elements in the quotient ring $R/\mathcal{J}$ where $\mathcal{J} = \langle \boldsymbol{g}^2 \rangle$. Furthermore, we setup the scheme in such a way that the homomorphic computation performed on the encoded values is preserved (mod $\mathcal{J}$). Note that even though the plaintext space has been changed to $R/\mathcal{J}$, the zero-test parameter is same as before. In particular, the zero-testing only checks if the encoded values is 0 (mod $\mathcal{I}$). Finally, q needs to be setup to support this functionality.

4.1 Our modified scheme

We will now describe our modified GGH scheme more formally. Just liked the GGH construction we use the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$.

Instance generation: $(\mathsf{params}, \mathsf{sparams}, \boldsymbol{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$**.** Chooses $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_\kappa$ uniformly at random in R_q such that for all $i \in [\kappa]$, $\|1/\boldsymbol{z}_i\| < n^2/q$[13] (in $K = \mathbb{Q}[X]/(X^n + 1)$) and $\boldsymbol{z}_i$ invertible in R_q (Lemma 3.7). Sample $\boldsymbol{g} \leftarrow D_{\mathbb{Z}^n,\sigma}$ with $\sigma = \sqrt{\lambda n}$ repeatedly till: (i) $\|\boldsymbol{g}\| \leq \sigma\sqrt{n}$ and $\boldsymbol{g}$ is invertible in R_q, (ii) $\|\boldsymbol{g}^{-1}\| \leq n^c$ (in K)[14] for an appropriate constant c, and (iii) $|R/\langle \boldsymbol{g} \rangle|$ is a prime $\geq 2^{O(n)}$. As argued in GGH such a $\boldsymbol{g}$ can be obtained in polynomially many trials.

Draw a "somewhat small" ring element $\boldsymbol{h} \leftarrow D_{\mathbb{Z}^n,\sqrt{q}}$, such that $\boldsymbol{h} \notin \mathcal{I}$ and set the zero-testing parameter as $\boldsymbol{p}_{zt} = [\boldsymbol{h} \prod_{i=1}^{\kappa} \boldsymbol{z}_i/\boldsymbol{g}]_q$. For security, $\boldsymbol{p}_{zt}$ needs to be sampled differently. This is described later.

The instance-generation procedure outputs the public parameters $\mathsf{params} = (n, q)$, the secret parameters $\mathsf{sparams} = (\boldsymbol{g}, \{\boldsymbol{z}_i\})$ and $\boldsymbol{p}_{zt}$.

Sampling level-zero encodings: $\boldsymbol{d} \leftarrow \mathsf{samp}(\mathsf{params})$**.** To sample a level-zero encoding of a random coset of $\mathcal{I}$, just draw a random short element in R, $\boldsymbol{d} \leftarrow D_{\mathbb{Z}^n,\sigma'}$, where $\sigma' = \sigma n\sqrt{\lambda}$ (for σ that was used to sample $\boldsymbol{g}$). The sampled value $\boldsymbol{d}$ corresponds to a random coset of $\mathcal{I}$ and the size of this level-zero encoding is bounded by $\sigma'\sqrt{n}$ (and we use this as our noise-bound for this encoding).

Our Encoding of $\boldsymbol{a} \in R/\mathcal{J}$ at level S: $\boldsymbol{u} \leftarrow \mathsf{enc}(\mathsf{params}, S, \boldsymbol{a})$**.** Sample $\boldsymbol{d} \leftarrow D_{\boldsymbol{a}+\mathcal{J},\sigma''}$, for some parameter σ'', and compute the encoding as $[\boldsymbol{d}/\prod_{i\in S} \boldsymbol{z}_i]_q$. For security, encoding needs to done differently. This is described later.

Adding and multiplying encodings. It is easy to see that the encoding as above is additively homomorphic, in the sense that adding encodings yields an encoding of the sum at the same level. This follows since if we have many short $\boldsymbol{c}_j$'s then their sum is still short, $\|\sum_j \boldsymbol{c}_j\| \ll q$, and therefore the sum $\boldsymbol{c} = \sum_j \boldsymbol{c}_j = [\sum_j \boldsymbol{c}_j]_q \in R_q$ belongs to the coset $\sum_j(\boldsymbol{c}_j + \mathcal{J})$. Hence, if we denote $\boldsymbol{u}_j = \boldsymbol{c}_j/\boldsymbol{z} \in R_q$ then each $\boldsymbol{u}_j$ is an encoding of the coset $\boldsymbol{c}_j + \mathcal{J}$, and the sum $[\sum_j \boldsymbol{u}_j]_q$ is of the form $\boldsymbol{c}/\boldsymbol{z}$ where $\boldsymbol{c}$ is still a short element in the sum of the cosets.

Moreover, since $\mathcal{J}$ is an ideal then multiplying up to κ encodings can be interpreted as an encoding of the product, by multiplying the denominators together. Namely, for $\boldsymbol{u}_j = \boldsymbol{c}_j/\boldsymbol{z}_j \in R_q$ as above, we have

$$\boldsymbol{u} = \left[\prod_{j=1}^{\kappa} \boldsymbol{u}_j\right]_q = \left[\frac{\prod_j \boldsymbol{c}_j}{\prod_j \boldsymbol{z}_j}\right]_q .$$

[13] This technical condition is needed for the secure sampling procedure.

[14] This technical condition is needed for the zero-test to work.

As long as the $\boldsymbol{c}_j$'s are small enough to begin with, we still have $\|\prod_j \boldsymbol{c}_j\| \ll q$, which means that $[\prod_j \boldsymbol{c}_j]_q = \prod_j \boldsymbol{c}_j$ (where the product $\prod_j \boldsymbol{c}_j$ is computed in R), hence $[\prod_j \boldsymbol{c}_j]_q$ belongs to the product coset $\prod_j (\boldsymbol{c}_j + \mathcal{J})$. Thus, if each $\boldsymbol{u}_j$ is an encoding of the coset $\boldsymbol{c}_j + \mathcal{J}$ with short-enough numerator, then their product is an encoding of the product coset.

Zero testing: $\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, \boldsymbol{u}_\kappa) \stackrel{?}{=} 0/1$. Recall that we are testing if the encoding is $0 \pmod{\mathcal{I}}$ (and not $0 \pmod{\mathcal{J}}$).

To test if a level-S encoding $\boldsymbol{u} = [\boldsymbol{c}/\prod_{i=1}^{\kappa} \boldsymbol{z}_i]_q$ is an encoding of $0 \pmod{\mathcal{I}}$, we just multiply it in R_q by $\boldsymbol{p}_{zt}$ and check whether the resulting element $\boldsymbol{w} = [\boldsymbol{p}_{zt} \cdot \boldsymbol{u}]_q$ is short (e.g., shorter than $q^{3/4}$). Namely, we use the test

$$\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, \boldsymbol{u}) \;=\; \begin{cases} 1 & \text{if } \|[\boldsymbol{p}_{zt}\boldsymbol{u}]_q\|_\infty < q^{3/4} \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

GGH argue that argue that encodings of $0 \pmod{\mathcal{I}}$ (such that the numerator is less than $q^{1/8}$) always pass the zero test and that encodings of non-zero cosets pass the zero test only with a negligible probability.

Secure encoding and zero-test. Following [GGH13a] we describe a modification of the encoding procedure and the zero-test generation procedure in order to defend against averaging attacks [GS02, NR06, DN12]. We describe how an encoding is done at level $\{1\}$ with noise bound $n^{O(1)}$. Encoding at any level-S can be done in an analogous manner with the noise $n^{O(|S|)}$.

Now we describe how to encode the $\boldsymbol{a}$ at level $\{1\}$. We assume that $\boldsymbol{a}$ is already reduced $(\mathrm{mod}\ \mathcal{J})$ and so $|\boldsymbol{a}| \leq \sigma^2 \cdot n^{3/2}$. Let $\mathcal{L}$ be the the fractional principal ideal $\langle \boldsymbol{a}/\boldsymbol{z}_1 \rangle$. Note that $\|\boldsymbol{a}/\boldsymbol{z}_1\| < \sigma^2 n^{7/2}/q$. And we can use the GPV sampling procedure to sample elements from $\mathcal{L}$ according to the Gaussian distribution $\boldsymbol{u} \leftarrow D_{\mathcal{L},s}$ with parameter $s = \sigma^2 n^4/q$ (say). Note that the $\boldsymbol{u}$ is of the form $\boldsymbol{u} = \boldsymbol{a}'/\boldsymbol{z}_1$ for some (integral) $\boldsymbol{a}' \in R$. Moreover we can bound the size of the $\boldsymbol{a}'$ by $\|\boldsymbol{a}'\| \leq \sqrt{n} \cdot \|\boldsymbol{u}\| \cdot \|\boldsymbol{z}_1\| < \sqrt{n} \cdot (\sigma^2 n^4/q) \cdot q\sqrt{n} = \sigma^2 n^5$. Next we map this elements to R_q and output $[\boldsymbol{a}'/\boldsymbol{z}_1]_q$, where the size of the numerator is bounded by $\sigma^2 n^5$. By an analogous tedious calculation one can show that encoding at level S corresponds to an encoding with noise bound $n^{O(|S|)}$.

The value $\boldsymbol{h}$ in $\boldsymbol{p}_{zt}$ is also sampled analogously. We use [GPV08] to prevent the attacker analyzing the zero-tester $\boldsymbol{h} \cdot \prod_{i=1}^{\kappa} \boldsymbol{z}_i/\boldsymbol{g}$ geometrically to extract useful information about $\boldsymbol{h}$. Roughly, once $\boldsymbol{g}$ and $\boldsymbol{z}_i$s are chosen, one chooses $\boldsymbol{h}$ according to an ellipsoid Gaussian of the same "shape" as $\boldsymbol{g}/\prod_{i=1}^{\kappa} \boldsymbol{z}_i$, so that the distribution of the zero-tester is a spherical Gaussian.

4.2 Setting the parameters

Note that the size of the numerator in an encoding provided at any level-S is $n^{O(|S|)}$ which is less than $2^{\lambda \cdot |S|}$, for large enough λ. Therefore the size of the numerator of an encoding at the highest level obtained by computing on the provided encodings will be less than $2^{\lambda \cdot \kappa} \cdot n^{O(\kappa)}$. Hence, as in [GGH13a] it suffices to set q larger than $2^{8\lambda \cdot \kappa} \cdot n^{O(\kappa)}$ for the zero-test to work.[15] In this section we provide the parameters for the scheme presented in Table 1.

[15] We note that the noise is an encoding at level-S is much smaller than $2^{\lambda \cdot |S|}$. Therefore, by careful calculation one can set q to be a much smaller value. This is better both for security and efficiency. However for the sake of simplicity we skip this optimization.

Parameter	Value Set
σ	$\sqrt{n\lambda}$
σ'	$\lambda n^{3/2}$
q	$\geq 2^{8\kappa\lambda} n^{O(\kappa)}$
n	$\geq \lambda^2 \log^2 q$

Table 1: Parameters for our graded encoding scheme.

5 Preliminaries for our Obfuscation Candidate

5.1 Branching Programs

Similar to previous works, we focus on obfuscating poly-length branching programs, which are known to be powerful enough to emulate all NC^1 circuits via Barrington's theorem [Bar86]. A branching program consists of a sequence of steps, where each step is defined by a pair of permutations. In each step the the program examines one input bit, and depending on its value the program chooses one of the permutations. Then computing on these permutations yields the result. Here we use the dual-input variant first introduced by Barak et al.[BGK+14].

Definition 5.1 (Dual-Input Oblivious Branching Program [BGK+14]) *A dual-input permutation matrix branching program of width d, length ℓ over m-bit inputs is given by a sequence,*

$$\mathsf{BP} = (\mathsf{inp}_1, \mathsf{inp}_2, \boldsymbol{A}_0, \{\boldsymbol{A}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{A}_{\ell+1})$$

where $\mathsf{inp}_1(\cdot), \mathsf{inp}_2(\cdot) : [\ell] \to [m]$ are functions such that $\mathsf{inp}_1(i)$ and $\mathsf{inp}_2(i)$ are the bits examined in step i, $\boldsymbol{A}_0 \in \{0,1\}^{1\times d} \setminus (0^d)^T$, $\boldsymbol{A}_{\ell+1} \in \{0,1\}^{d\times 1} \setminus 0^d$ and $\boldsymbol{A}_{i,b_1,b_2}$ are permutation matrices over $\{0,1\}^{d\times d}$. The output of the matrix branching program on an input $x \in \{0,1\}^m$ is given by:

$$\mathsf{BP}(x) = \begin{cases} 1 & \boldsymbol{A}_0 \left(\prod_{i\in[\ell]} \boldsymbol{A}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \boldsymbol{A}_{\ell+1} = 1 \\ 0 & \boldsymbol{A}_0 \left(\prod_{i\in[\ell]} \boldsymbol{A}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \boldsymbol{A}_{\ell+1} = 0 \end{cases}$$

A dual-input branching program is said to be input-oblivious *if the functions inp_1 and inp_2 are fixed and independent of the program being computed by $\boldsymbol{A}_0, \boldsymbol{A}_{i,b_1,b_2\,i\in[\ell],b_1,b_2}, \boldsymbol{A}_{\ell+1}$.*

Remark 5.2 *We will generalize the notion of permutation branching program to be over an arbitrary ring K instead of being over $\{0,1\}$. More formally, we have $\boldsymbol{A}_0, \{\boldsymbol{A}_{i,b_1,b_2}\}, \boldsymbol{A}_{\ell+1}$ to be elements in $K^{1\times d}, K^{d\times d}, K^{d\times 1}$ respectively. We have the correctness criterion modified as following:*

$$\mathsf{BP}(x) = \begin{cases} 1 & \boldsymbol{A}_0 \left(\prod_{i\in[\ell]} \boldsymbol{A}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \boldsymbol{A}_{\ell+1} \neq \mathbf{0} \\ 0 & \boldsymbol{A}_0 \left(\prod_{i\in[\ell]} \boldsymbol{A}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \boldsymbol{A}_{\ell+1} = \mathbf{0} \end{cases}$$

where $\mathbf{0}$ represents the additive identity in the ring K.

Remark 5.3 *A dual input branching program is called non-shortcutting if for any input $x \in \{0,1\}^n$*

$$\boldsymbol{A}_0 \times \boldsymbol{A}_{1,x_{\mathsf{inp}_1}(1),x_{\mathsf{inp}_2}(1)} \cdots \boldsymbol{A}_{\ell,x_{\mathsf{inp}_1}(\ell),x_{\mathsf{inp}_2}(\ell)} \neq \mathbf{0}^T$$
$$\boldsymbol{A}_{1,x_{\mathsf{inp}_1}(1),x_{\mathsf{inp}_2}(1)} \cdots \boldsymbol{A}_{\ell,x_{\mathsf{inp}_1}(\ell),x_{\mathsf{inp}_2}(\ell)} \times \boldsymbol{A}_{\ell+1} \neq \mathbf{0}$$

Notice that any permutation branching program is non-shortcutting since $\boldsymbol{A}_0 \neq (0^d)^T$, $\boldsymbol{A}_{\ell+1} \neq 0^d$ *and* $\boldsymbol{A}_{i,b_1,b_2}$ *are permutation matrices.*

Barrington [Bar86] showed that all circuits in NC^1 can be equivalently presented by a branching program of polynomial length.

Theorem 5.4 ([Bar86]) *For any depth-d, fan-in-2 boolean circuit* C*, there exists an oblivious branching program of width 5 and length at most* 4^d *that computes the same function as the circuit* C*.*

5.2 Straddling Sets

The straddling set system comprises of a system of sets satisfying specific combinatorial properties. Barak et al. in [BGK+14] defined these and provided a construction of such set systems.[16]

Definition 5.5 (Straddling Set System [BGK+14]) *A set system* $\mathbb{S} := \{S_{j,b}\}_{j\in[n],b\in\{0,1\}}$ *with* n *entries over an universe* $\mathbb{U}$ *is said to be straddling if*

- $\cup_{j\in[n]} S_{j,0} = \mathbb{U} = \cup_{j\in[n]} S_{j,1}$
- *For every non-empty sets* $C, D \subset \mathbb{S}$ *such that* C *and* D *contain only disjoint sets and* $\cup_{S\in C} S = \cup_{S\in D} S$ *then,*

  $$C = \cup_{j\in[n]} S_{j,b} \text{ and } D = \cup_{j\in[n]} S_{j,(1-b)}$$

  for some $b \in \{0,1\}$

We refer the reader to [BGK+14, Section 3] for a construction of a straddling set system.

6 Our Construction

In this section, we describe our candidate construction of indistinguishability obfuscation. We describe our construction only for the setting of NC^1 circuits. We note that the construction extends to general circuits using bootstrapping technique of [GGH+13b].

Input circuit to be obfuscated. Let R be the ring $\mathbb{Z}[X]/(X^n+1)$. Let $P : \{0,1\}^m \to \{0,1\}$ be the input circuit and

$$\mathsf{BP}_{\mathsf{main}} := \left\{\mathsf{inp}_1, \mathsf{inp}_2, \boldsymbol{P}_0, \{\boldsymbol{P}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{P}_{\ell+1}\right\}$$

denote the corresponding oblivious branching program over the ring R. In the remainder of the paper, for ease of notation, we use $\{\boldsymbol{P}_{i,b_1,b_2}\}$ to denote $\{\boldsymbol{P}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}$. Note that $\boldsymbol{P}_0, \{\boldsymbol{P}_{i,b_1,b_2}\}, \boldsymbol{P}_{\ell+1}$ belong to $R^{1\times 5}, R^{5\times 5}, R^{5\times 1}$ respectively.

Sampling the parameters for modified GGH multilinear maps. Generate the parameters for multilinear map $((n,q),(\boldsymbol{g},\{\boldsymbol{z}_i\}),\boldsymbol{p}_{zt}) \leftarrow \mathsf{InstGen}(1^\lambda, 1^\kappa)$, where $\kappa = 4\ell + 2$.

[16] Through the straddling sets were formally defined in [BGK+14] they were used implicitly in [GGH+13b].

Randomizing Program. We start with description of the randomizing program which computes the PRF. For an input $x \in \{0,1\}^m$ and a key $\psi \in \{0,1\}^\lambda$ let $C^\psi(x)$ be the circuit that outputs the first bit of $\mathsf{PRF}_\psi(x)$, where PRF is a pseudorandom function which on input any m bit string (with a λ bit key ψ) outputs λ pseudorandom bits.[17] Let the sequence of matrices $\{\boldsymbol{C}_0^\psi, \{\boldsymbol{C}_{i,b_1,b_2}^\psi\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{C}_{\ell+1}^\psi\}$ represent the oblivious input branching program of width 5 corresponding to the circuit C^ψ.

Next consider the following branching program[18] $\boldsymbol{B}_0^\psi, \{\boldsymbol{B}_{i,b_1,b_2}^\psi\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{B}_{\ell+1}^\psi$ of width $5s$ with $\psi_i = \mathsf{PRF}_\psi(i)$ for each $i \in \{1,\dots s\}$.

$$\boldsymbol{B}_0^\psi := \left(1\cdot\boldsymbol{C}_0^{\psi_1}, \cdots, 2^{s-1}\cdot\boldsymbol{C}_0^{\psi_s}\right) \quad \boldsymbol{B}_{i,b_1,b_2}^\psi := \begin{pmatrix} \boldsymbol{C}_{i,b_1,b_2}^{\psi_1} & & \\ & \ddots & \\ & & \boldsymbol{C}_{i,b_1,b_2}^{\psi_s} \end{pmatrix} \quad \boldsymbol{B}_{\ell+1}^\psi := \begin{pmatrix} \boldsymbol{C}_{\ell+1}^{\psi_1} \\ \vdots \\ \boldsymbol{C}_{\ell+1}^{\psi_s} \end{pmatrix} \tag{6.1}$$

where $\boldsymbol{B}_0^\psi \in R^{1\times 5s}, \boldsymbol{B}_{i,b_1,b_2}^\psi \in R^{5s\times 5s}$ and $\boldsymbol{B}_{\ell+1}^\psi \in R^{5s\times 1}$.

Let $\boldsymbol{e}_i$ be the element in the ring R corresponding to the i-th unit vector e_i in $\mathbb{Z}^n$.[19] Then define the branching programs $\boldsymbol{Q}_0^\chi, \{\boldsymbol{Q}_{i,b_1,b_2}^\chi\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{Q}_{\ell+1}^\chi$ of width $5ns$ with $\chi_i = \mathsf{PRF}_\chi(i)$ for each $i \in \{1,\dots n\}$:

$$\boldsymbol{Q}_0^\chi := (\boldsymbol{e}_1\cdot\boldsymbol{B}_0^{\chi_1}\ ,\ \cdots\ ,\ \boldsymbol{e}_n\cdot\boldsymbol{B}_0^{\chi_n}) \quad \boldsymbol{Q}_{i,b_1,b_2}^\chi := \begin{pmatrix} \boldsymbol{B}_{i,b_1,b_2}^{\chi_1} & & \\ & \ddots & \\ & & \boldsymbol{B}_{i,b_1,b_2}^{\chi_n} \end{pmatrix} \quad \boldsymbol{Q}_{\ell+1}^\chi := \begin{pmatrix} \boldsymbol{B}_{\ell+1}^{\chi_1} \\ \vdots \\ \boldsymbol{B}_{\ell+1}^{\chi_n} \end{pmatrix} \tag{6.2}$$

where $\boldsymbol{Q}_0^\chi \in R^{1\times 5ns}$, $\boldsymbol{Q}_{i,b_1,b_2} \in R^{5ns\times 5ns}$ and $\boldsymbol{Q}_{\ell+1}^\chi \in R^{5ns\times 1}$. The unspecified values in the above matrices are set to 0.

Now, without loss of generality, we will assume that

- Each of the oblivious branching programs described above have same length ℓ.
- Every step of the branching program inspects two different bits. That is for all $i \in [\ell]$ we have $\mathsf{inp}_1(i) \neq \mathsf{inp}_2(i)$.
- Every pair of bits is inspected in some step of the branching program. That is, for every pair of input bits $i_1, i_2 \in [n]$ such that $i_1 \neq i_2$ there exists $j \in [\ell]$ such that $(\mathsf{inp}_1(j), \mathsf{inp}_2(j)) = (i_1, i_2)$
- Every input bit is examined in exactly ℓ' steps. More precisely, if $\mathsf{ind}(j)$ denotes the set of steps that inspect input bit j:

  $$\mathsf{ind}(j) = \{i \in [\ell] : \mathsf{inp}_1(i) = j\} \cup \{i \in [\ell] : \mathsf{inp}_2(i) = j\}$$

  then we assume that for each $j \in [n]$ we have $|\mathsf{ind}(j)| = \ell'$.

[17] Note that for appropriate choice of the PRF the size of the circuit C is $\mathsf{poly}(\lambda)$ and it can be computed in NC^1.

[18] Here we abuse the terminology branching program slightly by referring to combinations of multiple branching programs that are input-aligned.

[19] Note that the PRF is computed bit-wise, hence the combined branching programs can only output the PRF value in integer ring $\mathbb{Z}^n$. However our encoding scheme encodes elements from the ring $R = \mathbb{Z}[X]/(X^n+1)$. Therefore appropriate multiplications with the ring elements $\boldsymbol{e}_i$'s are required to convert the PRF output in $\mathbb{Z}^n$ to a ring element.

It is straightforward to see that the above conditions are indeed without loss of generality, as they can be ensured by padding or inserting identity matrices whenever necessary without affecting the functionality. We keep it informal for simplicity.

Main Branching Program. Sample uniform random $\chi \leftarrow \{0,1\}^\lambda$ and random discrete gaussian samples $\boldsymbol{\alpha}_0, \{\boldsymbol{\alpha}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{\alpha}_{\ell+1} \leftarrow D_{R,\sigma'}$ for appropriate σ' (as stated in Table 1). Define the branching programs $\boldsymbol{A}_0, \{\boldsymbol{A}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{A}_{\ell+1}$ of width $d = 5ns + 5$:

$$\boldsymbol{A}_0 := \boldsymbol{\alpha}_0 \cdot (\boldsymbol{g}\cdot\boldsymbol{Q}_0^\chi \ , \ \boldsymbol{P}_0) \qquad \boldsymbol{A}_{i,b_1,b_2} := \boldsymbol{\alpha}_{i,b_1,b_2} \cdot \begin{pmatrix} \boldsymbol{Q}^\chi_{i,b_1,b_2} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{i,b_1,b_2} \end{pmatrix} \qquad \boldsymbol{A}_{\ell+1} := \boldsymbol{\alpha}_{\ell+1} \cdot \begin{pmatrix} \boldsymbol{Q}^\chi_{\ell+1} \\ \boldsymbol{P}_{\ell+1} \end{pmatrix} \tag{6.3}$$

where $\boldsymbol{A}_0 \in R^{1\times d}$, $\boldsymbol{A}_{i,b_1,b_2} \in R^{d\times d}$ and $\boldsymbol{A}_{\ell+1} \in R^{d\times 1}$ and $d = 5ns + 5$.

Kilian Randomization. Sample random matrices $\{\boldsymbol{R}_0, \cdots, \boldsymbol{R}_{\ell+1}\} \xleftarrow{\$} (D_{R,\sigma'})^{d\times d}$ and for all $i \in [\ell], b_1, b_2 \in \{0,1\}$ from appropriate discrete gaussian distribution and compute

$$\widetilde{\boldsymbol{A}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 \ ; \ \widetilde{\boldsymbol{A}}_{i,b_1,b_2} := \boldsymbol{R}^{adj}_{i-1} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i \ ; \ \widetilde{\boldsymbol{A}}_{\ell+1} := \boldsymbol{R}^{adj}_{\ell+1} \times \boldsymbol{A}_{\ell+1} \tag{6.4}$$

where $\boldsymbol{R}^{adj}$ is the adjoin of $\boldsymbol{R}$ in $R/\mathcal{I}$. We have $\widetilde{\boldsymbol{A}}_0 \in R^{1\times d}$, $\widetilde{\boldsymbol{A}}_{i,b_1,b_2} \in R^{d\times d}$ and $\widetilde{\boldsymbol{A}}_{\ell+1} \in R^{d\times 1}$.

Straddling Sets. For every $j \in [n]$, let $\mathbb{S}^j$ be a straddling set system with ℓ' entries over a set $\mathbb{U}_j$, such that the sets $\mathbb{U}_1, \ldots, \mathbb{U}_n$ are disjoint. Let $\mathbb{U} = \bigcup_{j\in[n]} \mathbb{U}_j$, and let U_s and U_t be sets such that $\mathbb{U}, U_s, U_t$ are disjoint. We associate the set system $\mathbb{S}^j$ with the j'th input bit. We index the elements of $\mathbb{S}^j$ by the steps of the branching program BP that inspect the j'th input. Namely,

$$\mathbb{S}^j = \left\{ S^j_{k,b} : k \in \mathsf{ind}(j), b \in \{0,1\} \right\},$$

for each $j \in [n]$. For every step $i \in [\ell]$, we denote by $S(i, b_1, b_2)$ the union of pairs of sets that are indexed by i:

$$\begin{aligned} S(i,b_1,b_2) &= S^{\mathsf{inp}_1(i)}_{i,b_1} \cup S^{\mathsf{inp}_2(i)}_{i,b_2} \\ S_0 &= U_s \\ S_{\ell+1} &= U_t \ . \end{aligned}$$

Note that $\kappa = |\mathbb{U} \cup U_s \cup U_t|$.

Obfuscation. Let $\boldsymbol{M}$ be a matrix over the ring R. Then $[\![\boldsymbol{M}]\!]_S$ denotes a matrix of the same dimensions where each entry is a modified GGH encoding[20] of the corresponding entry in $\boldsymbol{M}$ at the level S.

The obfuscation consists of encodings of the $\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_{i,b_1,b_2}, \widetilde{\boldsymbol{A}}_{\ell+1}$ at levels S_0, S_{i,b_1,b_2} and $S_{\ell+1}$.

- Compute $\left[\!\left[\widetilde{\boldsymbol{A}}_0\right]\!\right]_{S_0} \leftarrow \mathsf{enc}(\mathsf{params}, S_0, \widetilde{\boldsymbol{A}}_0)$, $\left[\!\left[\widetilde{\boldsymbol{A}}_{i,b_1,b_2}\right]\!\right]_{S(i,b_1,b_2)} \leftarrow \mathsf{enc}(\mathsf{params}, S_{i,b_1,b_2}, \widetilde{\boldsymbol{A}}_{i,b_1,b_2})$ for all $i \in [\ell], b_1, b_2 \in \{0,1\}$ and $\left[\!\left[\widetilde{\boldsymbol{A}}_{\ell+1}\right]\!\right]_{S_{\ell+1}} \leftarrow \mathsf{enc}(\mathsf{params}, S_{\ell+1}, \widetilde{\boldsymbol{A}}_{\ell+1})$.

[20] Note that this notations are different from the notations $[\cdot]$ used for hybrid graded encoding in the security proof in Sec. 8. Those notations are defined in Sec. 7.1.

- Output the obfuscation as:

$$\left\{ \left[\!\left[\widetilde{\boldsymbol{A}}_0 \right]\!\right]_{S_0}, \left\{ \left[\!\left[\widetilde{\boldsymbol{A}}_{i,b_1,b_2} \right]\!\right]_{S(i,b_1,b_2)} \right\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \left[\!\left[\widetilde{\boldsymbol{A}}_{\ell+1} \right]\!\right]_{S_{\ell+1}} \right\}.$$

Evaluation The evaluation works as follows:

- To evaluate $\mathcal{O}(\mathsf{BP})$ on some input $x \in \{0,1\}^m$ compute (in R_q):

$$\boldsymbol{u}_x := \left[\!\left[\widetilde{\boldsymbol{A}}_0 \right]\!\right]_{S_0} \cdot \left(\prod_{i=1}^{\ell} \left[\!\left[\widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}} \right]\!\right]_{S(i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)})} \right) \cdot \left[\!\left[\widetilde{\boldsymbol{A}}_{\ell+1} \right]\!\right]_{S_{\ell+1}} \tag{6.5}$$

- Output $1 - \mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, \boldsymbol{u}_x)$.

6.1 Correctness

In this section, we argue the correctness of our construction. Note that for arguing correctness it suffices to show that for all x, $\boldsymbol{u}_x$ is an encoding of $0 \pmod{\mathcal{I}}$ at the highest level $\mathbb{U}$ iff $P(x) = 0$. We will start by proving the following more general lemma that will be useful for the proof.

Lemma 6.1 *For every $x \in \{0,1\}^\lambda$ we have $\boldsymbol{u}_x$ (as in Equation 6.5) is an encoding of the coset*

$$\boldsymbol{c}_x \in \boldsymbol{\alpha}_x \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(x) + \boldsymbol{g} \cdot \boldsymbol{f}_x + P(x)) + \mathcal{J} \tag{6.6}$$

at level $\{1, \ldots \kappa\}$, where $\boldsymbol{\alpha}_x = \boldsymbol{\alpha}_0 \left(\prod_{i\in[\ell]} \boldsymbol{\alpha}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}} \right) \boldsymbol{\alpha}_{\ell+1}$ and for some $\boldsymbol{f}_x \in R$ that depends on the randomness used in obfuscation.

Proof From our construction recall that,

$$\boldsymbol{A}_0 := \boldsymbol{\alpha}_0 \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}_0^\chi \ , \ \boldsymbol{P}_0) \quad \boldsymbol{A}_{i,b_1,b_2} := \boldsymbol{\alpha}_{i,b_1,b_2} \cdot \begin{pmatrix} \boldsymbol{Q}^\chi_{i,b_1,b_2} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{i,b_1,b_2} \end{pmatrix} \quad \boldsymbol{A}_{\ell+1} := \boldsymbol{\alpha}_{\ell+1} \cdot \begin{pmatrix} \boldsymbol{Q}^\chi_{\ell+1} \\ \boldsymbol{P}_{\ell+1} \end{pmatrix}.$$

Note that,

$$\boldsymbol{A}_0 \times \left(\prod_{i\in[\ell]} \boldsymbol{A}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}} \right) \times \boldsymbol{A}_{\ell+1} = \boldsymbol{\alpha}_x \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(x) + P(x))$$

Finally we note that Kilian re-randomization and our encoding procedure introduces some multiples of $\boldsymbol{g}$ and $\boldsymbol{g}^2$. This proves the desired claim. ■

The fact that $\boldsymbol{u}_x$ is at level-$\{1, \ldots, \kappa\}$ follows by inspection and all we need to prove is that for all x, $\boldsymbol{c}_x = 0 \pmod{\mathcal{I}}$ iff $P(x) = 0$. Formally,

Lemma 6.2 *$\boldsymbol{c}_x = 0 \pmod{\mathcal{I}}$ (as in Equation 6.6) if and only if $P(x) = 0$.*

Proof From Lemma 6.1 we have that $\boldsymbol{c}_x \pmod{\mathcal{I}} = \boldsymbol{\alpha}_x \cdot P(x) \pmod{\mathcal{I}}$. Now we have two cases to consider:

- $P(x) = 0$: In this case, $\boldsymbol{c}_x \in \mathcal{I}$ and hence from zero test property we have: $\mathsf{isZero}(\mathsf{params}, \boldsymbol{p}_{zt}, \boldsymbol{u}_x) = 1$.

- $P(x) = 1$: In this case, we argue that $\boldsymbol{c}_x \notin \mathcal{I}$ with overwhelming probability and the result follows from the zero test property. Since $P(x) = 1$, $\boldsymbol{c}_x \in \mathcal{I}$ if and only if $\boldsymbol{\alpha}_x \in \mathcal{I}$. Since $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_{i,b_1,b_2}, \boldsymbol{\alpha}_{\ell+1}$ are sampled from $D_{R,\sigma'}$ and σ' is chosen (c.f. Table 1) to be sufficiently larger than the smoothing parameter of the lattice corresponding to $\mathcal{I}$, from Lemma 3.2 we have $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_{i,b_1,b_2}$ and $\boldsymbol{\alpha}_{\ell+1}$ are uniform in $R/\mathcal{I}$. Hence, probability that any one of $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_{i,b_1,b_2}$ or $\boldsymbol{\alpha}_{\ell+1}$ is $0 \pmod{\mathcal{I}}$ is negligible in λ. This along with the fact that $\mathcal{I}$ is a prime ideal, implies that for every $x \in \{0,1\}^\lambda$, $\boldsymbol{\alpha}_x \notin \mathcal{I}$ with overwhelming probability.

This concludes the proof. ■

7 Preliminaries for Security Arguments

7.1 The Hybrid Graded Encoding Model

In this section, we will describe our idealized model, which we call the *Hybrid Graded Encoding Model* where all parties have access to an oracle $\mathcal{M}$ implementing the graded encoding scheme. Informally, similar to [BGK+14], $\mathcal{M}$ will allow algebraic operations to be performed on encodings through so-called "handles" on the encodings. However, unlike [BGK+14], it will also allow arbitrary polynomial computation on the ring elements produced via "successful zero-tests," through a second type of handles.[21]

Similar to [BGK+14] we start by defining the hybrid graded encoding system.

Definition 7.1 (Hybrid Graded Encoding System) *Let $R = \mathbb{Z}[X]/X^n + 1$ be the $2n$-th cyclotomic ring of integers and $\boldsymbol{g} \in R$ be a "short" element in the ring such that $|R/\langle \boldsymbol{g} \rangle|$ is a prime of size $\omega(\mathsf{poly}(\lambda))$. Let $\mathbb{U}$ be a universal set. Then an encoding e of an element $\boldsymbol{t} \in R$ at the level $S \subseteq \mathbb{U}$ is denoted as $e = [\boldsymbol{t}]_S$. For any such encoding $e = [\boldsymbol{t}]_S$, the corresponding ring element $\boldsymbol{t}$ is called its representation and the set S its level. We define the following operations over the encodings.*

Addition: *Given two encodings $e_1 = [\boldsymbol{t}_1]_{S_1}$ and $e_2 = [\boldsymbol{t}_2]_{S_2}$ where $S_1 = S_2$, $e_1 + e_2$ is defined to be the encoding given by $[\boldsymbol{t}_1 + \boldsymbol{t}_2]_{S_1}$. Similarly, $e_1 - e_2$ is defined to be the encoding given by $[\boldsymbol{t}_1 - \boldsymbol{t}_2]_{S_1}$.*

Multiplication: *Given two encodings $e_1 = [\boldsymbol{t}_1]_{S_1}$ and $e_2 = [\boldsymbol{t}_2]_{S_2}$ where $S_1 \cap S_2 = \phi$, $e_1 \cdot e_2$ is defined to be the element given by $[\boldsymbol{t}_1 \cdot \boldsymbol{t}_2]_{S_1 \cup S_2}$.*

Ring Multiplication: *Given a ring element $\boldsymbol{a} \in R$ and an encoding $e = [\boldsymbol{t}]_S$, the ring multiplication $\boldsymbol{a} \cdot e$ is defined to be the encoding given by $e' = [\boldsymbol{a} \cdot \boldsymbol{t}]_S$.*[22]

Zero Testing: *For any encoding $e = [\boldsymbol{t}]_S$, it returns 1 if and only if:*

$$\boldsymbol{t} \pmod{\mathcal{I}} = 0 \text{ and } S = \mathbb{U}$$

[21] A reader familiar with [MSZ16] can note that this step is analogous to the type-2 query in that model.

[22] Note that we abuse the notation "$\cdot$" to denote both ring multiplication and multiplication between encodings.

We now proceed to describe the hybrid graded encoding model. Similar to [BGK+14] we consider a stateful oracle $\mathcal{M}$ mapping encodings to "generic" representations called handles. There are two types of handles that $\mathcal{M}$ generates: *encoding* handles that are corresponding to encodings and *ring* handles that are corresponding to the elements in the ring R (obtained after successful zero-tests). The handles are denoted by $\mathsf{H}_{\mathsf{Enc}}(e)$ for an encoding e and $\mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a})$ for any ring element $\boldsymbol{a} \in R$. We do not specify how the handles are generated. However, we require that the value of the handles, $\mathsf{H}_{\mathsf{Enc}}(e), \mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a})$ are independent of the corresponding encoding e and the corresponding ring element $\boldsymbol{a}$ respectively. The oracle maintains two tables L_{enc} and L_{rng} where L_{enc} stores encoding-handle pairs $(e, \mathsf{H}_{\mathsf{Enc}}(e))$ and similarly L_{rng} stores pairs of the form $(a, \mathsf{H}_{\mathsf{Rng}}(a))$ where $\mathsf{H}_{\mathsf{Rng}}(a)$ is a ring handle corresponding to ring element $\boldsymbol{a} \in R$. $\mathcal{M}$ provides the user with the following interfaces.

- **Initialization.** The oracle $\mathcal{M}$ is initialized with the parameters of the hybrid graded encoding system. Additionally, it is initialized with the encoding-handle table L_{enc} of initial encodings-handles pair and the ring-handle table L_{rng} with $\emptyset$. After $\mathcal{M}$ has been initialized, all subsequent calls to the initialization interfaces fail.

- **Algebraic operations.** Depending on the type of query it executes the following steps.

  - *Both are encoding handles:* Given two encoding handles $\mathsf{H}_{\mathsf{Enc}}(e_1), \mathsf{H}_{\mathsf{Enc}}(e_2)$ and an operation $\circ \in \{+, -, \cdot\}$, $\mathcal{M}$ first locates the relevant encodings $e_1 = [\boldsymbol{t}_1]_{S_1}$, $e_2 = [\boldsymbol{t}_2]_{S_2}$ in the handle table L_{enc}. If any of the input handles does not appear in the table L_{enc} (that is, if the handle was not previously generated by $\mathcal{M}$) the call to $\mathcal{M}$ fails. If the expression $e_1 \circ e_2$ is undefined (i.e., $S_1 \neq S_2$ for $\circ \in \{+, -\}$, or $S_1 \cap S_2 \neq \phi$ for $\circ \in \{\cdot\}$) the call fails. Otherwise, $\mathcal{M}$ generates a new encoding handle $\mathsf{H}_{\mathsf{Enc}}(e')$ for $e' = e_1 \circ e_2$. It appends the pair $(e', \mathsf{H}_{\mathsf{Enc}}(e'))$ into the table L_{enc} and returns $\mathsf{H}_{\mathsf{Enc}}(e')$.
  - *An encoding handle and a ring element:* Given a ring element $\boldsymbol{a} \in R$, an encoding handle $\mathsf{H}_{\mathsf{Enc}}(e)$ and a multiplication operation $\cdot$ first it checks if the encoding handle already exists in the corresponding table L_{enc}.[23] If it does not exist then this call fails. Otherwise, it computes the new encoding $e' = \boldsymbol{a} \cdot e$ via ring multiplication and generates the new handle $\mathsf{H}_{\mathsf{Enc}}(e')$. It appends the entry $(e', \mathsf{H}_{\mathsf{Enc}}(e'))$ into the table L_{enc} and outputs $\mathsf{H}_{\mathsf{Enc}}(e')$.

- **Zero testing.** Given a encoding-handle $\mathsf{H}_{\mathsf{Enc}}(e)$ as input, $\mathcal{M}$ first locates the corresponding encoding $e = [\boldsymbol{t}]_S$ in L_{enc}. If it is not found then (that is, if $\mathsf{H}_{\mathsf{Enc}}(e)$ was not previously generated by $\mathcal{M}$) then call to $\mathcal{M}$ fails. Otherwise, it performs zero-test on e. If the zero test fails, then this call fails. If it passes (i.e. returns 1) then recall from Definition 7.1 that $\boldsymbol{t} = 0 \mod \boldsymbol{g}$ which, in turn implies that $\boldsymbol{t}$ must be of the form $\boldsymbol{t} = \boldsymbol{a}'\boldsymbol{g}$. So it computes the ring element $\boldsymbol{a}' = \boldsymbol{t}/\boldsymbol{g}$, generates the corresponding ring handle $\mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a}')$, appends the pair $(\boldsymbol{a}', \mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a}'))$ into the table L_{rng} and outputs $\mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a}')$.

- **Post-zeroizing computation.** Given a non-zero polynomial p of bounded degree and a sequence of ring handles $\mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a}_1), \cdots, \mathsf{H}_{\mathsf{Rng}}(\boldsymbol{a}_t)$, $\mathcal{M}$ first locates the corresponding elements $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_t$ in the table L_{rng}. If any of them is not found in L_{rng} (that is not generated by

[23] Note that the only operation we allow is the multiplication. Note that for GGH construction (and for its modification that we consider) addition of a ring element to an encoding is not well-defined.

the above zero-test query) then call to $\mathcal{M}$ fails. Otherwise, $\mathcal{M}$ evaluates the polynomial $\widehat{\boldsymbol{a}} := p(\boldsymbol{a}_1, \cdots, \boldsymbol{a}_t)$. Then it checks if $\widehat{\boldsymbol{a}} = 0 \pmod{\mathcal{I}}$ and $\widehat{\boldsymbol{a}} \neq 0$. If the check fails then it returns 0. Otherwise, it returns 1. Furthermore, in this case $\mathcal{M}$ reveals its entire state including both lists L_{enc} and L_{rng} and the secret $\boldsymbol{g}$.[24]

Note that the construction does not need access to the post-zeroing computation. Only the attacker gets access to these queries.

Remark 7.2 *We note that one natural restriction that is implicitly placed on the attacker is that the attacker is not allowed to use the ring elements stored in the handle-table L_{rng} in multiplying with the encodings itself. This is a reasonable restriction because all ring elements generated after zero-test (the ones with corresponding handles in L_{rng}) are "large" and multiplying it with any encoding makes the numerator in that encoding large enough such that no zero-test can be performed on it.*

7.2 Virtual Black box Obfuscation in the Hybrid Graded Encoding Model

We now define the virtual black box obfuscation property in an idealized model where all algorithms have access to an oracle $\mathcal{M}$. Later we will prove that our construction achieves this definition in the hybrid graded encoding model in which $\mathcal{M}$ is an oracle as described above. As mentioned earlier, our construction doesn't need the post-zeroing computation and these queries are meant to provide the attacker with additional power.

Definition 7.3 ("Virtual Black-Box" Obfuscation in an $\mathcal{M}$-idealized model [BGK+14]) *For a (possibly randomized) oracle $\mathcal{M}$, and a circuit class $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$, we say that a uniform PPT oracle machine $\mathcal{O}$ is a "Virtual Black-Box" Obfuscator for $\{\mathcal{C}_\ell\}_{\ell\in\mathbb{N}}$ in the $\mathcal{M}$-idealized model, if the following conditions are satisfied:*

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

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

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

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

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

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

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

[24] Intuitively, if the adversary is able to query such a polynomial then it wins. Formally, this is captured in the model by making the oracle to output the entire state of the oracle.

7.3 The Sampling Lemma

Recall that, in our construction, a pseudorandom function is being computed in a specific manner. For our proof, we need the output of this function to be uniform over $\mathsf{mod}\,\mathcal{I}$. We prove that under appropriate choice of parameters this is indeed true.

Lemma 7.4 *Let $E_{\mathbb{Z}^n,s}$ be the distribution that outputs a random point in the $\{0, 2^s - 1\}^n$ and let $\mathcal{I} \subset \mathbb{Z}^n$ such that $diam(\mathcal{P}(\mathcal{I})) = poly(\lambda, n)$, where $\mathcal{P}(\mathcal{I})$ is the fundamental parallelepiped of $\mathcal{I}$ and $diam(\mathcal{P}(\mathcal{I}))$ is the maximum distance between any two points in $\mathcal{P}(\mathcal{I})$. The distribution $E_{\mathbb{Z}^n,s}\,\mathsf{mod}\,\mathcal{I}$ for any $s \geq \lambda$ is statistically close to uniform over $\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}$.*

Proof For any $\boldsymbol{a} \in \mathbb{Z}^n$, we define $\mathcal{P}(\mathcal{I}, \boldsymbol{a})$ to be the set of points obtained by adding $\boldsymbol{a}'$ to each point in $\mathcal{P}(\mathcal{I})$ where $\boldsymbol{a}' = \boldsymbol{a} - (\boldsymbol{a}\,\mathsf{mod}\,\mathcal{P}(\mathcal{I}))$. Let $\mathcal{S} = \{0, 2^s - 1\}^n$. Let us define a partition $\mathcal{S}_{\mathsf{int}}, \mathcal{S}_{\mathsf{bnd}}$ of $\mathcal{S}$ where

$$\mathcal{S}_{\mathsf{bnd}} := \{\boldsymbol{a} \in \mathcal{S} : \exists \boldsymbol{b} \in \mathcal{P}(\mathcal{I}, \boldsymbol{a}) \text{ s.t. } \boldsymbol{b} \notin \mathcal{S}\}$$

and $\mathcal{S}_{\mathsf{int}} = \mathcal{S} \setminus \mathcal{S}_{\mathcal{S}_{\mathsf{bnd}}}$. Let ξ be an arbitrary element in $\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}$.

Note that conditioned on $\boldsymbol{r} \in \mathcal{S}_{\mathsf{int}}$, $\boldsymbol{r}\,\mathsf{mod}\,\mathcal{I}$ is uniform in $\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}$. Therefore, the statistical distance between $E_{\mathbb{Z}^n,s}\,\mathsf{mod}\,\mathcal{I}$ and $U_{\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}}$ (uniform distribution over $\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}$) is

$$\begin{aligned} \frac{1}{2} \sum_{\xi \in \mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}} \left| \Pr_{\boldsymbol{r} \leftarrow \mathcal{S}}[\boldsymbol{r}\,\mathsf{mod}\,\mathcal{I} = \xi] - \Pr_{\boldsymbol{r} \leftarrow U_{\mathbb{Z}^n\,\mathsf{mod}\,\mathcal{I}}}[\boldsymbol{r} = \xi] \right| \\ \leq \frac{1}{2} \Pr_{\boldsymbol{r} \leftarrow \mathcal{S}}[\boldsymbol{r} \in \mathcal{S}_{\mathsf{bnd}}] \end{aligned} \tag{7.1}$$

and it suffices to argue that $\Pr_{\boldsymbol{r} \leftarrow \mathcal{S}}[\boldsymbol{r} \in \mathcal{S}_{\mathsf{bnd}}]$ is negligible. Let $\mathcal{S}^- := \{0, 2^s - 1 - diam(\mathcal{P}(\mathcal{I}))\}^n$. Note that for any $\boldsymbol{a} \in \mathcal{S}^-$ we have that $\mathcal{P}(\mathcal{I}, \boldsymbol{a}) \subset \mathcal{S}$. This follows since $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{P}(\mathcal{I}, \boldsymbol{a})$ we have $\|\boldsymbol{a} - \boldsymbol{b}\| \leq diam(\mathcal{P}(\mathcal{I}))$. Note that,

$$\frac{|\mathcal{S}_{\mathsf{bnd}}|}{|\mathcal{S}|} \leq \frac{|\mathcal{S}| - |\mathcal{S}^-|}{|\mathcal{S}|} \leq \frac{2^{sn} - (2^s - diam(\mathcal{P}(\mathcal{I})))^n}{2^{sn}} = 1 - \left(1 - \frac{diam(\mathcal{P}(\mathcal{I}))}{2^s}\right)^n \leq diam(\mathcal{P}(\mathcal{I}))n/2^s$$

which is negligible as $2^s \gg diam(\mathcal{P}(\mathcal{I}))n$. ■

8 The Security proof

In this section we prove that our construction is VBB-secure in the hybrid graded encoding model. In particular, we show that our construction as described in Sec. 6 achieves Def. 7.3 for all NC^1 circuits when $\mathcal{M}$ is the oracle in the hybrid graded encoding model. We start by describing our construction in that model. A formal description of the model is provided in Section 7.1.

8.1 Our construction in the Hybrid Graded Encoding model

Setup. Recall that we work in the $2n$-th cyclotomic ring over integers $R = \mathbb{Z}[X]/(X^n + 1)$ and consider the following ideals $\mathcal{I} = \langle \boldsymbol{g} \rangle$ and $\mathcal{J} = \langle \boldsymbol{g}^2 \rangle$ generated by $\boldsymbol{g}$ where $\boldsymbol{g} \in R$ is a "short" ring

element. We restate the following equations from our construction (Equations 6.3, 6.4 of Section 6):

$$\boldsymbol{A}_0 := \boldsymbol{\alpha}_0 \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}_0^{\chi} \ , \ \boldsymbol{P}_0) \quad \boldsymbol{A}_{i,b_1,b_2} := \boldsymbol{\alpha}_{i,b_1,b_2} \cdot \begin{pmatrix} \boldsymbol{Q}_{i,b_1,b_2}^{\chi} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{i,b_1,b_2} \end{pmatrix} \quad \boldsymbol{A}_{\ell+1} := \boldsymbol{\alpha}_{\ell+1} \cdot \begin{pmatrix} \boldsymbol{Q}_{\ell+1}^{\chi} \\ \boldsymbol{P}_{\ell+1} \end{pmatrix} \tag{8.1}$$

$$\widetilde{\boldsymbol{A}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 \ ; \ \widetilde{\boldsymbol{A}}_{i,b_1,b_2} := \boldsymbol{R}_{i-1}^{adj} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i \ ; \ \widetilde{\boldsymbol{A}}_{\ell+1} := \boldsymbol{R}_{\ell+1}^{adj} \times \boldsymbol{A}_{\ell+1} \tag{8.2}$$

where the $\boldsymbol{P}$ matrices are the parts of the main branching program that is to be obfuscated and $\boldsymbol{Q}$ matrices are parts of the auxiliary programs evaluating the PRF, $\boldsymbol{\alpha}$'s and the entries of the $\boldsymbol{R}$ matrices are random elements from discrete gaussian $D_{R,\sigma'}$ (for an appropriate σ' as in Table 1) and $\boldsymbol{R}^{adj}$'s are the adjoins of $\boldsymbol{R}$'s computed in $R/\mathcal{I}$.

We define:

$$\widetilde{\boldsymbol{B}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 + \boldsymbol{W}_0 \cdot \boldsymbol{g}^2 \ ; \ \widetilde{\boldsymbol{B}}_{i,b_1,b_2} := \boldsymbol{R}_{i-1}^{\mathsf{adj}} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i + \boldsymbol{W}_{i,b_1,b_2} \cdot \boldsymbol{g}^2 \ ; \ \widetilde{\boldsymbol{B}}_{\ell+1} := \boldsymbol{R}_{\ell+1}^{\mathsf{adj}} \times \boldsymbol{A}_{\ell+1} + \boldsymbol{W}_{\ell+1} \cdot \boldsymbol{g}^2 \tag{8.3}$$

for some matrices $\boldsymbol{W}_0, \{\boldsymbol{W}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{W}_{\ell+1}$ with entries from the ring R.[25] Let us denote the $\boldsymbol{A}, \widetilde{\boldsymbol{A}}$ matrices without the α's as:

$$\boldsymbol{C}_0 := (\boldsymbol{g} \cdot \boldsymbol{Q}_0^{\chi} \ , \ \boldsymbol{P}_0) \quad \boldsymbol{C}_{i,b_1,b_2} := \begin{pmatrix} \boldsymbol{Q}_{i,b_1,b_2}^{\chi} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{i,b_1,b_2} \end{pmatrix} \quad \boldsymbol{C}_{\ell+1} := \begin{pmatrix} \boldsymbol{Q}_{\ell+1}^{\chi} \\ \boldsymbol{P}_{\ell+1} \end{pmatrix} \tag{8.4}$$

$$\widetilde{\boldsymbol{C}}_0 := \boldsymbol{C}_0 \times \boldsymbol{R}_0 \ ; \ \widetilde{\boldsymbol{C}}_{i,b_1,b_2} := \boldsymbol{R}_{i-1}^{adj} \times \boldsymbol{C}_{i,b_1,b_2} \times \boldsymbol{R}_i \ ; \ \widetilde{\boldsymbol{C}}_{\ell+1} := \boldsymbol{R}_{\ell+1}^{adj} \times \boldsymbol{C}_{\ell+1} \tag{8.5}$$

Abusing notation slightly we will use $\{\boldsymbol{M}_{i,b_1,b_2}\}$ to denote the set $\{\boldsymbol{M}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}$ for any matrix $\boldsymbol{M}$ used in this section.

Initialization. The oracle $\mathcal{M}$ is initialized with the encoding parameters $R, \mathcal{I}, \boldsymbol{g}, \mathbb{U}$ and the following encodings:[26]

$$\left\{ \left[\widetilde{\boldsymbol{B}}_0\right]_{S_0}, \left\{ \left[\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\right]_{S(i,b_1,b_2)} \right\}, \left[\widetilde{\boldsymbol{B}}_{\ell+1}\right]_{S_{\ell+1}} \right\}$$

Obfuscation. Let $\mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_0\right]_{S_0}\right), \left\{\mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\right]_{S(i,b_1,b_2)}\right)\right\}, \mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_{\ell+1}\right]_{S_{\ell+1}}\right)$ be the corresponding encoding handles returned by $\mathcal{M}$. Then we define the output of the obfuscation algorithm to be:

$$\mathcal{O}^{\mathcal{M}} := \left\{\mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_0\right]_{S_0}\right), \left\{\mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\right]_{S(i,b_1,b_2)}\right)\right\}, \mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_{\ell+1}\right]_{S_{\ell+1}}\right)\right\}$$

[25] In the actual construction the matrices $\boldsymbol{W}_0, \{\boldsymbol{W}_{i,b_1,b_2}\}_{i\in[\ell],b_1,b_2\in\{0,1\}}, \boldsymbol{W}_{\ell+1}$ are generated from the encoding procedure of our multilinear map construction (see Sec. 4 for details). Note that we do not explicitly mention the actual distributions from which $\boldsymbol{W}_0, \boldsymbol{W}_{i,b_1,b_2}, \boldsymbol{W}_{\ell+1}$ are sampled as that is not important in order to prove security.

[26] Note that, here we use the hybrid encoding notations, that are different from the modified GGH encoding notations used in the construction in Sec. 6.

Evaluation. To evaluate the obfuscation on input $x \in \{0,1\}^m$ we compute (using oracle calls to $\mathcal{M}$):

$$\mathsf{H}_x^{\mathsf{op}} := \mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_0\right]_{S_0}\right) \times \left(\prod_{i=1}^{\ell} \mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}} i, x_{\mathsf{inp}_1(i)}, x_{\mathsf{inp}_2(i)}\right]_{S(i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)})}\right)\right) \times \mathsf{H}_{\mathsf{Enc}}\left(\left[\widetilde{\boldsymbol{B}}_{\ell+1}\right]_{S_{\ell+1}}\right)$$

and zero test on the output encoding handle $\mathsf{H}_x^{\mathsf{op}}$. If the call to zero test outputs 0 then we return 1. Else, we return 0. The correctness of the evaluation follows from Lemma 6.1, 6.2 in a straightforward manner.

8.2 Proof of Security in the Hybrid Encoding Model

Formally we prove the following theorem.

Theorem 8.1 *Let P be an arbitrary circuit in NC^1. Let $\mathcal{O}^{\mathcal{M}}$ be the obfuscation of P in the Hybrid Graded Encoding Model with respect to the oracle $\mathcal{M}$. Then for every probabilistic polynomial time (PPT) adversary $\mathcal{A}^{\mathcal{M}}$, there exists a poly time simulator Sim such that the following holds:*

$$\mathcal{A}^{\mathcal{M}}(\mathcal{O}^{\mathcal{M}}(P)) \stackrel{c}{\approx} \mathsf{Sim}^P(1^{|P|})$$

Road Map. First in Section 8.2.1 we describe our simulator. In section 8.2.2 we prove the correctness of the simulation of the zero-testing queries. Finally, in section 8.2.3 we prove the correctness of the simulation of the post-zeroizing queries.

8.2.1 Description of the Simulator

We remark that many of our techniques are analogous to [BGK+14, BMSZ16, AGIS14]. However, we diverge in simulating the zero-testing and post zeroizing queries.

To prove VBB security, for every adversary $\mathcal{A}$, we construct a PPT simulator Sim, given $1^{|P|}$, the description of an adversary $\mathcal{A}$ and oracle access to the circuit P, is able to simulate the view of $\mathcal{A}$ that is computationally close to its view in the real-world execution. Sim starts by emulating the obfuscation algorithm $\mathcal{O}$. Note that, Sim is not given the actual circuit P as input but only oracle access to P. Therefore, it does not know the $\boldsymbol{P}_{i,b_1,b_2}$ matrices that are used in the obfuscation. Hence, similar to the earlier works, instead of initializing the actual matrices, Siminitializes $\mathcal{M}$ with formal variables corresponding to each entry of the matrices.

Towards that, first we extend the definition of an encoding (defined formally in Sec. 7.1) such that its representation can also be a formal variable. When performing an operation $\circ$ on encodings $e_1 = [\boldsymbol{t}_1]_{S_1}, e_2 = [\boldsymbol{t}_2]_{S_2}$ for formal variables $\boldsymbol{t}_1, \boldsymbol{t}_2$, the value of the resulting element $e_1 \circ e_2$ is naturally defined as $[\boldsymbol{t}_1 \circ \boldsymbol{t}_2]_S$(for an appropriate S and assuming the operation is defined for such levels). We represent formal expressions as arithmetic circuits, thereby guaranteeing that the representation size remains polynomial in the security parameter.

Sim emulates $\mathcal{O}$ by instantiating with appropriate $R, \mathcal{I}, \mathbb{U}$ as in the obfuscation construction (Section 6) and creates random handles corresponding to the encodings:

$$\left\{\left[\widetilde{\boldsymbol{B}}_0\right]_{S_0}, \left\{\left[\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\right]_{S(i,b_1,b_2)}\right\}, \left[\widetilde{\boldsymbol{B}}_{\ell+1}\right]_{S_{\ell+1}}\right\}$$

where the encodings themselves are represented by corresponding formal variables.

Sim proceeds to emulate the execution of the adversary $\mathcal{A}$ on input handles corresponding to the formal variables defined above. It is straightforward to observe that when $\mathcal{A}$ makes an oracle call that is neither a zero test nor a post-zeroizing query, Sim *perfectly* emulates $\mathcal{M}$'s answer as the distribution of handles generated during the simulation and during the real execution are identical and the simulated obfuscation consists only of handles (as opposed to the actual encodings).

So the only things left to explain are how Sim correctly simulates the zero test queries and post zeroizing queries.

Simulation of Zero Testing Queries. When the adversary $\mathcal{A}$ makes a zero-test query with a handle h, Sim looks up the corresponding polynomial p (represented as an arithmetic circuit) in the handle table. In this work, we would be interested in the polynomial p modulo $\mathcal{J}$ (denoted by $p_{\mathcal{J}}$).[27] Like in previous works [BGK+14, BMSZ16, AGIS14] we first decompose $p_{\mathcal{J}}$ into the so-called "single-input" elements over the quotient ring $R/\mathcal{J}$. Formally we define single-input elements as follows.

Definition 8.2 *A single input element for an input x is a polynomial $p_x \in R/\mathcal{J}[\widetilde{\boldsymbol{C}}_0, \widetilde{\boldsymbol{C}}_{1,x_{\mathsf{inp}_1(1)},x_{\mathsf{inp}_2(1)}}, \cdots, \widetilde{\boldsymbol{C}}_{\ell,x_{\mathsf{inp}_1(\ell)},x_{\mathsf{inp}_2(\ell)}}, \widetilde{\boldsymbol{C}}_{\ell+1}]$ that is expressible as*

$$p_x = \sum_{(j_0,i_1,j_1,\cdots,i_{\ell+1})} c_{j_0,\cdots,i_{\ell+1}} \cdot (\widetilde{\boldsymbol{C}}_0)_{1,j_0} \cdot (\widetilde{\boldsymbol{C}}_{1,x_{\mathsf{inp}_1(1)},x_{\mathsf{inp}_2(1)}})_{i_1,j_1} \cdots (\widetilde{\boldsymbol{C}}_{\ell+1})_{i_{\ell+1},1} \bmod \mathcal{J}$$

where $c_{j_0,\cdots,i_{\ell+1}} \in R/\mathcal{J}$ and $j_0, i_1, \cdots, i_{\ell+1} \in [d]$.

The following proposition, adapted from [BGK+14, BMSZ16] to our setting, states that for any polynomial p which an adversary submits as a zero testing query, the corresponding $p_{\mathcal{J}}$ can be efficiently decomposed into *polynomial* number of single-input elements.

Proposition 8.1 ([BGK+14, Claim 2] [BMSZ16, Lemma 5.3]) *For every polynomial $p \in R[\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1}]$, $p_{\mathcal{J}} := p \bmod \mathcal{J}$ can be efficiently decomposed into a sum $p_{\mathcal{J}} = \sum_{x \in D} \boldsymbol{\alpha}_x \cdot p_x$ where p_x is a single input element for the input x, $\boldsymbol{\alpha}_x = \boldsymbol{\alpha}_0 \cdot \left(\prod_i \boldsymbol{\alpha}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \cdot \boldsymbol{\alpha}_{\ell+1}$ and D is a set containing polynomially (in λ) many input x.*

Sim's strategy to answer $\mathcal{A}$'s zero-test queries is described in detail in Figure 1.

Remark 8.3 *In this section whenever we mention about sampling a uniform random value from the ring $R/\mathcal{I}$ we implicitly mean that this is done by sampling a random value from the discrete gaussian $D_{R,\sigma'}$ for an appropriate σ' (as in Table 1).*

Simulating post zeroizing queries. For every post zero testing query that the adversary $\mathcal{A}$ gives Sim returns 0.

[27] Notice that, in $\bmod \mathcal{J}$, any term with a $\boldsymbol{g}^2$ multiplier (namely the $\boldsymbol{W}$ matrices in the Eqn. 8.3) vanishes.

Input: A polynomial p over $(\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1})$.

1. Let $p_{\mathcal{J}} := p \bmod \mathcal{J}$. Decompose $p_{\mathcal{J}}$ into polynomially many single input elements as per Lemma 8.1. Express $p_{\mathcal{J}}$ as:
$$p_{\mathcal{J}} = \sum_{x \in D} \boldsymbol{\alpha}_x \cdot p_x \bmod \mathcal{J}$$
where p_x is a single input element for the input x and $\boldsymbol{\alpha}_x = \boldsymbol{\alpha}_0 \cdot \left(\prod_i \boldsymbol{\alpha}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \boldsymbol{\alpha}_{\ell+1}$.

2. For every $x \in D$ do the following:

   (a) Choose random matrices $\widetilde{\boldsymbol{D}}_0, \{\widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{D}}_{\ell+1}$ from $R/\mathcal{I}$ and evaluate the polynomial $p_x \bmod \mathcal{I}$ (over $R/\mathcal{I}$) on these values. Let Val_x be the evaluated output ($\bmod \mathcal{I}$). If $\mathsf{Val}_x = 0$ then choose a new $x' \in D$ among the remaining ones, if no such x' remains then go to Step 3. Otherwise if $\mathsf{Val}_x \neq 0$ then go to the next step.

   (b) Query the oracle P to learn the output of P on input x. If $P(x) = 1$ then return 0. Otherwise if $P(x) = 0$ then go to the next step.

   (c) Choose $\widetilde{\boldsymbol{D}}_0, \{\widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}$ uniformly at random from $R/\mathcal{I}$. Let $\boldsymbol{v} = (v_1 \cdots v_d)$ be the row vector corresponding to $\widetilde{\boldsymbol{D}}_0 \times \prod_{i\in\ell} \widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}$. Then choose $w_2, \cdots, w_d$ uniformly at random from $R/\mathcal{I}$ and define:
$$\widetilde{\boldsymbol{D}}_{\ell+1} = \begin{pmatrix} -\sum_{i=2}^{d} v_i w_i \\ v_1 w_2 \\ \vdots \\ v_1 w_d \end{pmatrix} \tag{8.6}$$
   Evaluate the polynomial $p_x \bmod \mathcal{I}$ (over $R/\mathcal{I}$) on these (randomly) chosen values. Let Eval_x be the evaluation (in $\bmod \mathcal{I}$). Now if $\mathsf{Eval}_x \neq 0$ then return 0. Otherwise continue the loop with a new $x' \in D$. If no such x' remains then go to Step 3.

3. If for every $x \in D$, $\mathsf{Val}_x = 0$ (computed in Step 2a) then create a ring handle corresponding to $0 \in R$ and return the handle. Otherwise, in the case when $\mathsf{Eval}_x = 0$ for some $x \in D$, create a ring handle to the formal variable r_p such that $p = r_p \boldsymbol{g}$ (which is guaranteed by the construction) denoted by r_p and return the handle.

Figure 1: Sim's zero testing strategy

8.2.2 Correctness of Zero Testing in the Simulation

We start by proving some general results on the structure of "allowable" polynomials. These results will be used to prove simulation correctness. Before that we introduce some notation:

Notation. Let $\boldsymbol{A}_0, \boldsymbol{A}_1, \cdots, \boldsymbol{A}_{\ell+1}, \boldsymbol{Q}_0^{\chi}, \boldsymbol{Q}_1^{\chi}, \cdots, \boldsymbol{Q}_{\ell+1}^{\chi}, \boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_{\ell+1}$ be such that $\boldsymbol{A}_i = \boldsymbol{A}_{i,b_1,b_2}$, $\boldsymbol{Q}_i^{\chi} = \boldsymbol{Q}_{i,b_1,b_2}^{\chi}$, $\boldsymbol{P}_i = \boldsymbol{P}_{i,b_1,b_2}$ for some choice of b_1, b_2 and for all $i \in [\ell]$ (c.f. Equations 8.1, 8.2). For the sake of proving Theorem 8.7, we will assume that $\boldsymbol{A}_0, \boldsymbol{A}_1, \cdots, \boldsymbol{A}_{\ell+1}$, $\boldsymbol{Q}_0^{\chi}, \boldsymbol{Q}_1^{\chi}, \cdots, \boldsymbol{Q}_{\ell+1}^{\chi}$, $\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_{\ell+1}$ have entries from an arbitrary ring K (i.e Theorem 8.7 is agnostic to the underlying ring that we are working with), more precisely, $\widetilde{\boldsymbol{A}}_0 \in K^{d\times 1}, \widetilde{\boldsymbol{A}}_1 \in K^{d\times d}, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1} \in K^{1\times d}$. We define $\boldsymbol{R}_{-1} = \boldsymbol{R}_{\ell+2} = 1$. Then for all $i \in \{0, \cdots, \ell+1\}$ we have:

$$\widetilde{\boldsymbol{A}}_i = \boldsymbol{R}_{i-1}^{\mathsf{adj}} \times \boldsymbol{A} \times \boldsymbol{R}_i$$

Let us now define the notion of an allowable polynomial which would correspond to polynomials on a "single-input" x.

Definition 8.4 *A polynomial $p \in K[\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}]$ is said to be allowable if it can be expressed as*

$$p = \sum_{(j_0, i_1, j_1, \cdots, i_{\ell+1})} c_{j_0,\cdots,i_{\ell+1}} \cdot (\widetilde{\boldsymbol{A}}_0)_{1,j_0} \cdot (\widetilde{\boldsymbol{A}}_1)_{i_1,j_1} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1}$$

for some $c_{j_0,\cdots,i_{\ell+1}} \in K$ where $j_0, i_1, \cdots, i_{\ell+1} \in [d]$

Remark 8.5 *In the future references to allowable polynomials we do not refer to the explicit bounds on $j_0, i_1, \cdots, i_{\ell+1}$.*

We now state and prove a theorem regarding the structure of allowable polynomials over $(\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1})$.

Remark 8.6 *The theorem stated below is similar to the Theorem 4.2 in [BMSZ16]. The only difference is that we consider conditions 1, 2 to be on the $\boldsymbol{P}$ matrices instead of the $\boldsymbol{A}$ matrices. Concretely, our conditions state that $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} \neq 0$ and $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} = 0$ instead of stating $\boldsymbol{A}_0 \times \boldsymbol{A}_1 \cdots \times \boldsymbol{A}_{\ell+1} \neq 0$ and $\boldsymbol{A}_0 \times \boldsymbol{A}_1 \cdots \times \boldsymbol{A}_{\ell+1} = 0$ respectively. Moreover, our "non-short cutting" condition in the statement of the Theorem is based on $\boldsymbol{P}$ matrices rather than $\boldsymbol{A}$ matrices.*

Theorem 8.7 (Adapted from [BMSZ16, Theorem 4.2]) *Consider an allowable polynomial $p \in K[\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}]$ such that after making the substitutions*

$$\widetilde{\boldsymbol{A}}_0 \to \boldsymbol{A}_0 \times \boldsymbol{R}_0 \ , \ \left\{\widetilde{\boldsymbol{A}}_i \to \boldsymbol{R}_{i-1}^{\mathsf{adj}} \times \boldsymbol{A}_i \times \boldsymbol{R}_i\right\}_{i\in[\ell]} \text{ for all } i \in [\ell] \ , \ \widetilde{\boldsymbol{A}}_{\ell+1} \to \boldsymbol{R}_{\ell}^{\mathsf{adj}} \times \boldsymbol{A}_{\ell+1},$$

p is identically zero over $\{\boldsymbol{R}_k\}_{k\in\{0,\cdots,\ell\}}$. Moreover assume that,

$$\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell} \neq \boldsymbol{0}$$

and

$$\boldsymbol{P}_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} \neq \mathbf{0}^T$$

Then the following is true:

1. *If* $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} \neq 0$ *then* p *is identically zero as a polynomial over its formal variables, namely* $(\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1})$.
2. *If* $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} = 0$ *then* p *can be expressed as a constant multiple of* $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \widetilde{\boldsymbol{A}}_1 \times \cdots \widetilde{\boldsymbol{A}}_{\ell+1}$ *where* γ *is a constant in the ring* K.

Proof We prove the claim by induction on ℓ.

Base case: $\ell = -1$.

There is just one matrix namely $\widetilde{\boldsymbol{A}}_0$ matrix with a single entry $\boldsymbol{a}$ where $\boldsymbol{a}$ is a formal variable and there are no $\{\boldsymbol{R}_k\}$ matrices (since $\boldsymbol{R}_{-1} = \boldsymbol{R}_1 = \boldsymbol{I}$ in this case). Then any allowable polynomial p is of the form $c \cdot \boldsymbol{a}$. Since we are given that p is identically 0 over the non-existent $\{\boldsymbol{R}_k\}$ matrices, we infer that $c \cdot \boldsymbol{a} = 0$. This implies that either $c = 0$ or $\boldsymbol{a} = 0$. If $\boldsymbol{a} \neq 0$ then $c = 0$ and this corresponds to Item 1. In the other case, p is trivially a multiple of the matrix product. Thus we have proved the assertion for the base case.

Inductive Hypothesis. We assume that the hypothesis is true for $\ell - 1$.

Inductive step: We prove for the case ℓ assuming the hypothesis is true for $\ell - 1$. We assume that $\boldsymbol{A}_0$ has all non-zero entries and later show that this can be assumed without loss of generality. Any allowable polynomial over $(\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1})$ can be expressed as:

$$p = \sum_{(j_0, i_1, j_1, \cdots, i_{\ell+1})} c_{j_0, \cdots, i_{\ell+1}} \cdot (\widetilde{\boldsymbol{A}}_0)_{1,j_0} \cdot (\widetilde{\boldsymbol{A}}_1)_{i_1,j_1} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1}$$

Note that $(\widetilde{\boldsymbol{A}}_0)_{1,j_0} = \sum_m (\boldsymbol{A}_0)_{1,m} \cdot (\boldsymbol{R}_0)_{m,j_0}$ and $(\widetilde{\boldsymbol{A}}_1)_{i_1,j_1} = \sum_k (\boldsymbol{R}_0^{\mathsf{adj}})_{i_1,k} \cdot (\boldsymbol{A}_1 \cdot \boldsymbol{R}_1)_{k,j_1}$.

Hence,

$$p = \sum_{j,i,m,k} \alpha'_{j,i,k} \cdot (\boldsymbol{A}_0)_{1,m} \cdot (\boldsymbol{R}_0)_{m,j} \cdot (\boldsymbol{R}_0^{\mathsf{adj}})_{i,k}$$

where

$$\alpha'_{j,i,k} = \sum_{j_1, \cdots, i_{\ell+1}} c_{j,i,j_1,i_2,\cdots,i_{\ell+1}} (\boldsymbol{A}_1 \times \boldsymbol{R}_1)_{k,j_1} \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,j_2} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1}$$

Observe that:

$$(\boldsymbol{R}_0^{\mathsf{adj}})_{i,k} = \sum_{\sigma:\sigma(i)=k} \mathsf{sign}(\sigma) \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t} \right)$$

Hence, we can express p as:

$$p = \sum_{j,i,m,\sigma} \mathsf{sign}(\sigma) \cdot \alpha'_{j,i,\sigma(i)} \cdot (\boldsymbol{A}_0)_{1,m} \cdot (\boldsymbol{R}_0)_{m,j} \cdot \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t} \right) \tag{8.7}$$

We are given that p is identically 0 over the matrices $\{\boldsymbol{R}_k\}$. In particular p is identically 0 over the variables corresponding to $\boldsymbol{R}_0$ entries. In Equation 8.7, we collect the terms corresponding to p that are coefficients of the product $(\boldsymbol{R}_0)_{m,j} \cdot \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t}\right)$ and equate them to 0.

We now give details on how the products $(\boldsymbol{R}_0)_{m,j} \cdot \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t}\right)$ are formed. This process can be thought of as choosing a permutation σ and selecting entries according to the permutation. That is, for each column t choose the entry in the row given by $\sigma(t)$. Then for some i, we unselect the selected entry $(\boldsymbol{R}_0)_{\sigma(i),i}$ and instead select some other entry given by $(\boldsymbol{R}_0)_{m,j}$. Depending on the concrete values of i, j, m, σ we get the different products and thus we consider the cases described below.

- $m = \sigma(i), j = i$: This corresponds to re-selecting the un-selected entry from column i. Given the final list of selected entries, we observe that σ gets fixed but i can vary. The reason behind this is that reselecting an unselected entry does not depend on which column the entry was unselected. Given $\sigma, j = i, m = \sigma(i)$, we determine the coefficients of these products to be:

$$\sum_i \alpha'_{i,i,\sigma(i)} \cdot (\boldsymbol{A}_0)_{1,\sigma(i)} = 0 \text{ for all } \sigma \tag{8.8}$$

- $j \neq i$ and $m \neq \sigma(i)$: This corresponds to after unselecting an entry from column i, reselecting an entry from a different row and a column. Given the final list of selected entries, we can determine the unselected entry as the only one that shares no row or column with any another entry. Also, we can determine the reselected entry as the one that shares both a row and a column with another selected entry. This observations help us deduce that given the list of selected entries the permutation σ, i, $j \neq i$ and $m \neq \sigma$ are completely determined. Hence, there does not exist another (i', j', m', σ') that leads to the same product $(\boldsymbol{R}_0)_{m,j} \cdot \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t}\right)$. Thus, the coefficients of these products to be:

$$\alpha'_{j,i,\sigma(i)} \cdot (\boldsymbol{A}_0)_{1,m} = 0 \text{ for all } i,\ j \neq i,\ m,\ \sigma \text{ where } \sigma(i) \neq m$$

  Given any i, $j \neq i$, m and $s \neq m$ we pick a σ such that $\sigma(i) = s$. Since we are given that $(\boldsymbol{A}_0)_{1,m} \neq 0$ for all m and $d > 1$ we get:

$$\alpha'_{j,i,s} = 0 \text{ for all } i, j \neq i \text{ and for all } s \tag{8.9}$$

- $j = i$ and $m \neq \sigma(i)$: This means that there exists a row (which is given by m) that has two entries and the row corresponding to $\sigma(i)$ does not have any entry. Let (i', σ', m') be some other triplet that leads to the same product $(\boldsymbol{R}_0)_{m,j} \cdot \left(\prod_{t \neq i} (\boldsymbol{R}_0)_{\sigma(t),t}\right)$. We first observe that $m = m'$ as otherwise the product will correspond to different rows having two entries and hence they cannot be equal. Similarly, we can argue that $\sigma(i) = \sigma'(i')$ as otherwise different rows will have no entries selected and hence the product cannot be equal. Given the final selection of entries corresponding to (i, σ, m) it is possible to determine $m = m'$ as the only row with two selected entries. Furthermore, it is possible to determine $\sigma(i) = \sigma'(i)$ as the only row with no selected entry. Additionally, i is one of the selected columns in row m. Let us call the other selected column as i''. Apart from, i and i'' all other selected entries correspond to selection given by the permutation given by σ. Furthermore, it is straightforward to argue

that $\sigma(t) = \sigma'(t)$ for all $t \notin \{i, i''\}$ (as otherwise the products would be different). Let use consider following two cases:

- $i = i'$: In this case, since $\sigma(i) = \sigma'(i') = \sigma'(i)$. Hence, σ and σ' are equal to on all but one entry and since σ and σ' are permutations it must be the case that $\sigma = \sigma'$. Hence, we can infer that $(i, \sigma, m) = (i', \sigma', m')$.
- $i \neq i'$. In this case, we deduce that $i'' = i'$ (recall that i'' is the other selected column in row m). This means $\sigma(i') = m$. Analogously, we can argue that $\sigma'(i) = m' = m$. Since σ and σ' are permutations we can deduce from the fact that (i, σ, m) and (i', σ', m') lead to the same product that $\sigma' = \sigma \circ (i, i')$ where (i, i') is the transposition swapping i and i'. Since σ and σ' differ by a transposition they differ in their signs.

Thus, the coefficients of these products in this case correspond to:

$$(\alpha'_{i,i,\sigma(i)} - \alpha'_{i',i',\sigma(i)}) \cdot (\boldsymbol{A}_0)_{1,m} = 0 \text{ for all } i,\, i' \neq i,\, \sigma \text{ where } \sigma(i) \neq m$$

Given any $i, i', s \neq m$ we pick a σ such that $\sigma(i) = s$. Since we are given that $(\boldsymbol{A}_0)_{1,m} \neq 0$ for all m we get:

$$\alpha'_{i,i,s} = \alpha'_{i',i',s} \text{ for all } i, i' \text{ and for all } s \tag{8.10}$$

Combining Equation 8.10, 8.8 we get:

$$\sum_s \alpha'_{i,i,s} (\boldsymbol{A}_0)_{1,s} = 0 \tag{8.11}$$

- $j \neq i$ and $m = \sigma(i)$: The coefficients correspond to linear combinations of $\alpha_{j,i,s}$ for $i \neq j$ which we already know to be 0.

Consider the following polynomial $p_{j,i,s}$ for i, j where $j \neq i$ and s:

$$p_{j,i,s} = \alpha'_{j,i,s} = \sum_{j_1, \cdots, i_{\ell+1}} c_{j,i,j_1,i_2,\cdots,i_{\ell+1}} (\boldsymbol{A}_1 \times \boldsymbol{R}_1)_{s,j_1} \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,j_2} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1}$$

We first note that $\alpha'_{j,i,s}$ is an allowable polynomial over $\{(\boldsymbol{A}_1)_s \times \boldsymbol{R}_1, \widetilde{\boldsymbol{A}}_2, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}\}$. From Equation 8.9, we have $\alpha'_{j,i,s}$ is identically 0 for all s, j, i where $j \neq i$. We also observe conditions of the theorem that $\boldsymbol{P}_1 \times \boldsymbol{P}_2 \cdots \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}$. Hence, there is at least one t such that $(\boldsymbol{P}_1)_t \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} \neq 0$. We can also observe that $(\boldsymbol{P}_1)_t \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_\ell \neq \boldsymbol{0}^T$ and $\boldsymbol{P}_2 \times \boldsymbol{P}_3 \cdots \times \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}$. Hence, there exists a t' such that the polynomial $\alpha'_{j,i,t'}$ satisfies the conditions required by Item 1. Therefore, by induction hypothesis we get:

$$c_{j,i,j_1,i_2,\cdots,i_{\ell+1}} = 0 \text{ for all } i, j \neq i,\, j_1, \cdots, i_{\ell+1} \tag{8.12}$$

We consider another polynomial β'_i for all i:

$$\begin{aligned} \beta'_i = \sum_s \alpha'_{i,i,s} (\boldsymbol{A}_0)_{1,s} &= \sum_{s, j_1, \cdots, i_{\ell+1}} c_{i,i,j_1,i_2,\cdots,i_{\ell+1}} (\boldsymbol{A}_0)_{1,s} (\boldsymbol{A}_1 \times \boldsymbol{R}_1)_{s,j_1} \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,j_2} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1} \\ &= \sum_{j_1, \cdots, i_{\ell+1}} c_{i,i,j_1,i_2,\cdots,i_{\ell+1}} (\boldsymbol{A}_0 \times \boldsymbol{A}_1 \times \boldsymbol{R}_1)_{1,j_1} \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,j_2} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1} \end{aligned} \tag{8.13}$$

Let us denote $\boldsymbol{A}'_1 = \boldsymbol{A}_0 \times \boldsymbol{A}_1$ and $\widetilde{\boldsymbol{A}}'_1 = \boldsymbol{A}'_1 \times \boldsymbol{R}_1$. We correspondingly get $\boldsymbol{P}'_1 = \boldsymbol{P}_0 \times \boldsymbol{P}_1$. Then the above polynomial given in Equation 8.13 is allowable over $\widetilde{\boldsymbol{A}}'_1, \widetilde{\boldsymbol{A}}_2, \cdots \widetilde{\boldsymbol{A}}_{\ell+1}$. Moreover, from Equation 8.8 that $\beta'_i = 0$. We now have two cases to consider:

- $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \boldsymbol{P}_{\ell+1} \neq 0$. This means that $\boldsymbol{P}'_1 \times \boldsymbol{P}_2 \cdots \boldsymbol{P}_{\ell+1} \neq 0$. We also observe that $\boldsymbol{P}'_1 \times \boldsymbol{P}_2 \cdots \boldsymbol{P}_\ell \neq \boldsymbol{0}^T$ and $\boldsymbol{P}_2 \times \boldsymbol{P}_3 \cdots \times \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}$. Hence, the polynomial β'_i satisfies the conditions of Item 1. Therefore, by induction hypothesis we get, that (β'_i) is identically 0 over its formal variables,

$$c_{i,i,j_1,i_2,\cdots,i_{\ell+1}} = 0 \text{ for all } i, j_1, \cdots, i_{\ell+1} \tag{8.14}$$

  Combining Equations 8.12 and 8.14 we get p is identically 0.

- $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} = 0$: This means that $\boldsymbol{P}'_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} = 0$. We observe that $\boldsymbol{P}'_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_\ell = \boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_\ell \neq \boldsymbol{0}^T$ and $\boldsymbol{P}_2 \times \boldsymbol{P}_3 \cdots \times \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}$ (as otherwise $\boldsymbol{P}_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} = \boldsymbol{0}$ contradicting the condition given in the theorem statement) Hence, the polynomial β'_i satisfies the conditions of Item 2. Therefore, by induction hypothesis we get:

$$(\beta'_i) = (\gamma_i) \cdot \widetilde{\boldsymbol{A}}'_1 \times \widetilde{\boldsymbol{A}}_2 \cdots \times \widetilde{\boldsymbol{A}}_{\ell+1}$$

  As a result from Equation 8.13 we get,

$$c_{i,i,j_1,i_2,\cdots,i_{\ell+1}} = 0 \text{ if } j_k \neq i_{k+1} \text{ for any } k \tag{8.15}$$

$$c_{i,i,i_2,i_2,\cdots,i_{\ell+1}} = \gamma_i \tag{8.16}$$

  The Equations 8.15, 8.16 enforce the condition that β'_i is a matrix product. We already know from Equation 8.12 that

$$c_{j,i,j_1,i_2,\cdots,i_{\ell+1}} = 0 \text{ for all } i, j \neq i, j_1, \cdots, i_{\ell+1}$$

  We also know as result of Equations 8.15 and 8.16 that

$$\begin{aligned}
\alpha'_{i,i,s} &= \gamma_i \sum_{i_2,\cdots,i_{\ell+1}} (\boldsymbol{A}_1\boldsymbol{R}_1)_{s,i_2} \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,i_3} (\widetilde{\boldsymbol{A}})_{i_{\ell+1},1} \\
&= \gamma_i \sum_{i_3,\cdots,i_{\ell+1}} (\boldsymbol{A}_1)_s \times \boldsymbol{R}_1 \cdot (\widetilde{\boldsymbol{A}}_2)_{i_2,i_3} (\widetilde{\boldsymbol{A}})_{i_{\ell+1},1} \\
&= \gamma_i((\boldsymbol{A}_1)_s \times \boldsymbol{A}_3 \cdots \times \boldsymbol{A}_{\ell+1}) \\
&= \gamma_i(\boldsymbol{A}_1 \cdots \boldsymbol{A}_{\ell+1})_{s,1} \text{ for all } i, s
\end{aligned}$$

  Also we know that $\alpha'_{i,i,s} = \alpha'_{i',i',s}$ for all $i \neq i'$. Therefore, we have for all s,

$$\gamma_i(\boldsymbol{A}_1 \cdots \boldsymbol{A}_{\ell+1})_{s,1}) = \gamma_{i'}(\boldsymbol{A}_1 \cdots \boldsymbol{A}_{\ell+1})_{s,1})$$

Hence, we have $\gamma_i = \gamma$ since there exists an t such that $(\boldsymbol{P}_1 \cdots \boldsymbol{P}_{\ell+1})_{t,1} \neq 0$. Therefore, we get

$$c_{i,i,i_2,i_2,\cdots,i_{\ell+1}} = \gamma \text{ for all } i, i_2, i_3, \cdots, i_{\ell+1} \tag{8.17}$$

Combining Equations 8.15 and 8.17 we get:

$$p = \gamma \cdot \boldsymbol{C}_0 \cdots \boldsymbol{C}_{\ell+1}$$

We still need to show the proof goes through even if $\boldsymbol{A}_0$ has zero entries. We are also given that $\boldsymbol{A}_0$ is not identically zero (otherwise the matrix $\boldsymbol{P}_0$ would not satisfy the conditions of theorem). Given a non-zero vector we can construct an invertible matrix $\boldsymbol{B}$ such that $\boldsymbol{A}_0^\star = \boldsymbol{A}_0 \times \boldsymbol{B}$ is non-zero in every coordinate.

We now define $\boldsymbol{A}_1^\star := \boldsymbol{B}^{-1} \times \boldsymbol{A}_1$, $\boldsymbol{R}_0^\star := \boldsymbol{B}^{-1} \times \boldsymbol{R}_0$, $\widetilde{\boldsymbol{A}}_0^\star := \boldsymbol{A}_0^\star \times \boldsymbol{R}^\star{}_0$ and $\widetilde{\boldsymbol{A}}_1^\star := \boldsymbol{R}^{\star\mathsf{adj}}_0 \cdot \boldsymbol{A}_1^\star \cdot \boldsymbol{R}_1$. Now the set of matrices $\boldsymbol{A}^\star{}_0, \boldsymbol{A}_1^\star, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_{\ell+1}$ satisfy the set of constraints given by the statement of Theorem 8.7. In particular, they satisfy that $\boldsymbol{P}_0^\star \times \boldsymbol{P}_1^\star \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_\ell = \boldsymbol{P}_0 \times \boldsymbol{P}_1 \times \cdots \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}^T$ and $\boldsymbol{P}_1^\star \times \boldsymbol{A}_2 \cdots \times \boldsymbol{A}_{\ell+1} = \boldsymbol{B}^{-1} \times \boldsymbol{P}_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}$. We observe that p is allowable over $\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}$ if and only if it is allowable over $\widetilde{\boldsymbol{A}_0^\star}, \widetilde{\boldsymbol{A}^\star}_1, \widetilde{\boldsymbol{A}}_2, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}$. We can also relate the polynomial p over $\boldsymbol{R}_0, \cdots, \boldsymbol{R}_{\ell+1}$ to the a polynomial over $\boldsymbol{R}^\star{}_0, \boldsymbol{R}_1, \cdots, \boldsymbol{R}_{\ell+1}$ as a linear combination over the $\boldsymbol{R}_1$ variables. As a result, we can conclude that p is identically zero over $\{\boldsymbol{R}_k\}$ matrices if and only if it is identically zero over $\boldsymbol{R}^\star{}_0, \boldsymbol{R}_1, \cdots, \boldsymbol{R}_{\ell+1}$. Hence, we can invoke the Theorem to work for the set of matrices given by $\widetilde{\boldsymbol{A}_0^\star}, \widetilde{\boldsymbol{A}^\star}_1, \widetilde{\boldsymbol{A}}_2, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1}$ using the same polynomial p and arrive at the desired conclusion. ■

Now, recall from Equation 8.2,8.3 that:

$$\widetilde{\boldsymbol{A}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 \;;\; \widetilde{\boldsymbol{A}}_{i,b_1,b_2} := \boldsymbol{R}_{i-1}^{adj} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i \;;\; \widetilde{\boldsymbol{A}}_{\ell+1} := \boldsymbol{R}_{\ell+1}^{adj} \times \boldsymbol{A}_{\ell+1} \tag{8.18}$$

$$\widetilde{\boldsymbol{B}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 + \boldsymbol{W}_0 \cdot \boldsymbol{g}^2 \;;\; \widetilde{\boldsymbol{B}}_{i,b_1,b_2} := \boldsymbol{R}_{i-1}^{\mathsf{adj}} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i + \boldsymbol{W}_{i,b_1,b_2} \cdot \boldsymbol{g}^2 \;;\; \widetilde{\boldsymbol{B}}_{\ell+1} := \boldsymbol{R}_{\ell+1}^{\mathsf{adj}} \times \boldsymbol{A}_{\ell+1} + \boldsymbol{W}_{\ell+1} \cdot \boldsymbol{g}^2 \tag{8.19}$$

We define $\widetilde{\boldsymbol{B}}_i = \widetilde{\boldsymbol{B}}_{i,b_1,b_2}$ for $i \in [\ell]$ and for some choice of b_1, b_2. We analogously define $\widetilde{\boldsymbol{A}}_i := \widetilde{\boldsymbol{A}}_{i,b_1,b_2}$ for all $i \in [\ell]$ and for the same choice of b_1, b_2 that was used to select $\widetilde{\boldsymbol{B}}_i$. Note that for every $i \in \{0, \cdots, \ell+1\}$, the entries of $\widetilde{\boldsymbol{B}}_i, \widetilde{\boldsymbol{A}}_i$ are from the specific ring R (instead of any arbitrary ring). We consider allowable polynomials over $(\widetilde{\boldsymbol{B}}_0, \widetilde{\boldsymbol{B}}_1, \cdots, \widetilde{\boldsymbol{B}}_{\ell+1})$ to be of the form:

$$p = \sum_{(j_0, i_1, j_1, \cdots, i_{\ell+1})} d_{j_0,\cdots,i_{\ell+1}} \cdot (\widetilde{\boldsymbol{B}}_0)_{1,j_0} \cdot (\widetilde{\boldsymbol{B}}_1)_{i_1,j_1} \cdots (\widetilde{\boldsymbol{B}}_{\ell+1})_{i_{\ell+1},1}$$

for some $d_{j_0,\cdots,i_{\ell+1}} \in R$ which is also presented succinctly as $p \in R[\widetilde{\boldsymbol{B}}_0, \widetilde{\boldsymbol{B}}_1, \cdots, \widetilde{\boldsymbol{B}}_{\ell+1}]$. Clearly, we can have

$$p_{\mathcal{J}} = \sum_{(j_0, i_1, j_1, \cdots, i_{\ell+1})} c_{j_0,\cdots,i_{\ell+1}} \cdot (\widetilde{\boldsymbol{A}}_0)_{1,j_0} \cdot (\widetilde{\boldsymbol{A}}_1)_{i_1,j_1} \cdots (\widetilde{\boldsymbol{A}}_{\ell+1})_{i_{\ell+1},1} \bmod \mathcal{J}$$

where each $c_{j_0,\cdots,i_{\ell+1}} \in R/\mathcal{J}$. Now, the following corollary explicitly shows that if we instantiate the ring K in Theorem 8.7 with the specific ring $R/\mathcal{J}$ then the same result follows.

Corollary 8.8 *Consider an allowable polynomial* $p \in R[\widetilde{\boldsymbol{B}}_0, \widetilde{\boldsymbol{B}}_1, \cdots, \widetilde{\boldsymbol{B}}_{\ell+1}]$. *Let* $p_{\mathcal{J}} = p \bmod \mathcal{J}$ *be such that after making the substitutions* $\widetilde{\boldsymbol{A}}_0 \to \boldsymbol{A}_0 \times \boldsymbol{R}_0$, $\left\{\widetilde{\boldsymbol{A}}_i \to \boldsymbol{R}_{i-1}^{\mathsf{adj}} \times \boldsymbol{A}_i \times \boldsymbol{R}_i\right\}_{i\in[\ell]}$, $\widetilde{\boldsymbol{A}}_{\ell+1} \to \boldsymbol{R}_\ell^{\mathsf{adj}} \times \boldsymbol{A}_{\ell+1}$, $p_{\mathcal{J}}$ *is identically* $0 \bmod \mathcal{J}$ *over* $\{\boldsymbol{R}_k\}_{k\in\{0,\cdots,\ell\}}$. *Moreover assume that,*

$$\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_\ell \neq \boldsymbol{0} \bmod \mathcal{J}$$

and

$$\boldsymbol{P}_1 \times \boldsymbol{P}_2 \cdots \times \boldsymbol{P}_{\ell+1} \neq \boldsymbol{0}^T \bmod \mathcal{J}$$

then the following is true:

1. *If* $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} \neq 0 \bmod \mathcal{J}$ *then* $p_{\mathcal{J}}$ *is identically* $0 \bmod \mathcal{J}$ *as a polynomial over its formal variables, namely* $(\widetilde{\boldsymbol{A}}_0, \widetilde{\boldsymbol{A}}_1, \cdots, \widetilde{\boldsymbol{A}}_{\ell+1})$.
2. *If* $\boldsymbol{P}_0 \times \boldsymbol{P}_1 \cdots \times \boldsymbol{P}_{\ell+1} = 0 \bmod \mathcal{J}$ *then* $p_{\mathcal{J}}$ *can be expressed as* $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \widetilde{\boldsymbol{A}}_1 \times \cdots \widetilde{\boldsymbol{A}}_{\ell+1} \bmod \mathcal{J}$ *where* γ *is a constant in the ring* $R/\mathcal{J}$.

Proof Follows directly from Theorem 8.7 where K is instantiated with $R/\mathcal{J}$. ■

We set up some notation for the rest of the proof.

Some notations. In the remainder of the security proof, $\widetilde{\boldsymbol{A}}_0, \{\widetilde{\boldsymbol{A}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{A}}_{\ell+1}$ refers to formal variables. We refer to evaluation of a polynomial $p_{\mathcal{J}} \in R/\mathcal{J}[\widetilde{\boldsymbol{A}}_0, \{\widetilde{\boldsymbol{A}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{A}}_{\ell+1}]$ as the value (in $R/\mathcal{J}$) the polynomial $p_{\mathcal{J}}$ takes when the formal variables $\{\widetilde{\boldsymbol{A}}\}$ are substituted with the values as given in Equation 8.2 modulo $\mathcal{J}$. We shall succinctly refer the evaluation of $p_{\mathcal{J}}$ as $\mathsf{Eval}(p_{\mathcal{J}})$ where we implicitly assume the formal variables takes the values as given in Equation 8.2 modulo $\mathcal{J}$. Note that all the evaluations of a polynomial is over $R/\mathcal{J}$. We also denote $\mathsf{Eval}(p_x)$ and $\mathsf{Eval}(p)$ to denote evaluation of the respective polynomials where the formal variables again take values as given in Equation 8.4 and Equation 8.3 respectively.

Now, we now argue that the adversary's view in the real world execution of the zero test queries is computationally close to the simulated view. Formally we prove the following lemma.

Lemma 8.9 *For every polynomial* $p \in R[\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1}]$ *(defined in Equation 8.3) let* $p_{\mathcal{J}} = p \bmod \mathcal{J}$. *Then we have,*

1. $\mathsf{Eval}(p) \bmod \mathcal{I} \neq 0 \Leftrightarrow \mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0 \Leftrightarrow \mathsf{Sim}$ *returns* 0 *with overwhelming probability.*
2. *Let* $\mathsf{H}_{\mathsf{Rng}}(r_{\mathsf{Sim}})$ *denote the handle returned by* Sim[28] *if* $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} = 0$. p *is identically 0 if and only if* $r_{\mathsf{Sim}} = 0$.

Proof $\mathsf{Eval}(p) \bmod \mathcal{I} \neq 0 \Leftrightarrow \mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0$ can be easily seen. The remaining part of Step 1 of the statement follows from Claims 8.12, 8.19. Step 2 of this lemma follows from Claims 8.20. We prove the claims next which finishes the proof. ■

We prove the following claim regarding the structure of a single-input allowable polynomial which would be useful to prove Claim 8.12.

[28] Notice that here r_{Sim} denotes either the ring element $0 \in R$ or the formal variable r_p over the ring R as defined in Fig. 1.

Claim 8.10 ([BMSZ16]) *For any x, let p_x be an allowable polynomial in $K[\widetilde{\boldsymbol{A}}_0, \{\widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}, \widetilde{\boldsymbol{A}}_{\ell+1}]$ where K is an arbitrary ring such that after making the substitution*

$$\widetilde{\boldsymbol{A}}_{\ell+1} = \begin{pmatrix} -\sum_{i=2}^{d} v_i w_i \\ v_1 w_2 \\ \vdots \\ v_1 w_d \end{pmatrix}$$

p_x is identically 0 where $(v_1, \cdots, v_d) = \widetilde{\boldsymbol{A}}_0 \times \prod_{i\in\ell} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}$, then p_x is of the form $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{A}}_{\ell+1}$ for some $\gamma \in K$.

Proof We recall the proof from [BMSZ16]. p_x being identically 0 after the substitution given in claim means that p_x becomes identically 0 after the substitution

$$(\widetilde{\boldsymbol{A}}_{\ell+1})_{1,1} \leftarrow \frac{-\sum_{i=2}^{d} v_i (\widetilde{\boldsymbol{A}}_{\ell+1})_{i,1}}{v_1}$$

If this substitution gives the zero polynomial then it means that the polynomial is divisible by

$$(\widetilde{\boldsymbol{A}}_{\ell+1})_{1,1} + \frac{\sum_{i=2}^{d} v_i (\widetilde{\boldsymbol{A}}_{\ell+1})_{i,1}}{v_1}$$

Since p is a polynomial, we can remove v_1 in the denominator and conclude that in fact

$$v_1(\widetilde{\boldsymbol{A}}_{\ell+1})_{1,1} + \sum_{i=2}^{d} v_i (\widetilde{\boldsymbol{A}}_{\ell+1})_{i,1} = \sum_{i=1}^{d} v_i (\widetilde{\boldsymbol{A}}_{\ell+1})_{i,1}$$

divides p_x. But the polynomial $\sum_{i=1}^{d} v_i (\widetilde{\boldsymbol{A}}_{\ell+1})_{i,1}$ is exactly the matrix product polynomial. Since p_x is allowable it is of the form $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{A}}_{\ell+1}$ where $\gamma \in K$. ■

In the following corollary we explicitly state that the above property holds when K is instantiated with the specific ring $R/\mathcal{I}$.

Corollary 8.11 *For any x, let $p_x \bmod \mathcal{I}$ be an allowable polynomial in $R/\mathcal{I}[\widetilde{\boldsymbol{A}}_0, \{\widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}, \widetilde{\boldsymbol{A}}_{\ell+1}]$ such that after making the substitution*

$$\widetilde{\boldsymbol{A}}_{\ell+1} = \begin{pmatrix} -\sum_{i=2}^{d} v_i w_i \\ v_1 w_2 \\ \vdots \\ v_1 w_d \end{pmatrix} \bmod \mathcal{I} \tag{8.20}$$

$p_x \bmod \mathcal{I}$ is identically $0 \bmod \mathcal{I}$ where $(v_1, \cdots, v_d) = \widetilde{\boldsymbol{A}}_0 \times \prod_{i\in\ell} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}$, then p_x is of the form $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{A}}_{\ell+1} \bmod \mathcal{I}$ where $\gamma \in R/\mathcal{I}$.

Next we prove Claim 8.12 which implies "⇒" direction of Step 1 of Lemma 8.9.

Claim 8.12 *For every allowable polynomial* $p \in R[\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1}]$ *let* $p_{\mathcal{J}} := p \bmod \mathcal{J}$. *If* $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0$ *then* Sim *returns* 0 *with overwhelming probability.*

Proof Since we are given that $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0$, there exists at least one $x \in D$ such that $\mathsf{Eval}(p_x) \bmod \mathcal{I}$ is not zero. In particular, this implies that $p_x \bmod \mathcal{I}$ as a formal polynomial (i.e $\in R/\mathcal{I}[\widetilde{\boldsymbol{A}}_0, \{\widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{A}}_{\ell+1}]$) is not identically 0. We have two cases to consider:

- $P(x) = 1$: In this case, the simulator evaluates $p_x \bmod \mathcal{I}$ on randomly chosen $\widetilde{\boldsymbol{D}}_0$, $\{\widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{D}}_{\ell+1}$ from $R/\mathcal{I}$ and outputs 0 if evaluation of $p_x \bmod \mathcal{I}$ is not 0 on these inputs. Since $p_x \bmod \mathcal{I}$ is not identically 0 as a polynomial and it has $\mathsf{poly}(\lambda)$ degree (which is ensured by our multilinear structure), evaluation of $p_x \bmod \mathcal{I}$ on the random inputs is not 0 with overwhelming probability by Schwartz-Zippel Lemma. Hence, in this case Sim outputs 0 with overwhelming probability.
- $P(x) = 0$: Since we are given that $\mathsf{Eval}(p_x) \bmod \mathcal{I}$ is not zero we infer that $p_x \bmod \mathcal{I}$ is not of the form $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{A}}_{\ell+1} \bmod \mathcal{I}$ where $\gamma \in R/\mathcal{I} \setminus 0$. This means that by Corollary 8.11 we have that $p_x \bmod \mathcal{I}$ is not identically 0 after making the substitution in Equation 8.20. Hence, except with negligible probability over the random choice of $\widetilde{\boldsymbol{D}}_0, \{\widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{D}}_{\ell+1}$ and $w_2, \cdots, w_d$ we have from Schwartz-Zippel lemma that evaluation of p_x on inputs $\widetilde{\boldsymbol{D}}_0, \{\widetilde{\boldsymbol{D}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{D}}_{\ell+1}$ is not 0 over $R/\mathcal{I}$. Hence, Sim outputs 0 with overwhelming probability.

■

We now set the machinery to prove "⇐" of Step 1 of Lemma 8.9.

Claim 8.13 *Let* $x \in D$ *be such that* $\mathsf{Eval}(p_x) \bmod \mathcal{I} \neq 0$. *Then with overwhelming probability over the choices of* $\boldsymbol{\alpha}_0, \{\boldsymbol{\alpha}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}, \boldsymbol{\alpha}_{\ell+1}$ *we have* $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0$

Proof Since for at least one $x \in D$ it holds that, $\mathsf{Eval}(p_x) \bmod \mathcal{I} \neq 0$, we can see that (from the algebraic independence of $\boldsymbol{\alpha}_x$) $p_{\mathcal{J}} \bmod \mathcal{I}$ is not identically 0 with overwhelming probability over the random choice of $\boldsymbol{\alpha}_x$. Moreover, we note that since $\boldsymbol{\alpha}$'s are sampled from $D_{R,\sigma'}$ where σ' is larger than the smoothing parameter of the lattice $\mathcal{I}$, $\boldsymbol{\alpha}_0 \bmod \mathcal{I}, \{\boldsymbol{\alpha}_{i,b_1,b_2}\} \bmod \mathcal{I}, \boldsymbol{\alpha}_{\ell+1} \bmod \mathcal{I}$ are uniform in $R/\mathcal{I}$ (from Lemma 3.2). Hence by Schwartz-Zippel lemma we have that $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} \neq 0$ with overwhelming probability. ■

Claim 8.14 *If* $p_x = \gamma \cdot \widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{B}}_{\ell+1}$ *where* $\gamma \in R$ *then* $p_x \bmod \mathcal{I} = \gamma' \cdot \widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{B}}_{\ell+1} \bmod \mathcal{I}$ *where* $\gamma' \in R/\mathcal{I}$.

Proof It is straightforward to see this Claim. ■

Next we present three claims, which intuitively follow from the observation that if $\mathcal{A}$ obtains a non-zero element in the ideal $\mathcal{I}$ he would be able to win the game without making further zero testing queries.

Claim 8.15 *If* p_x *after substitution as given in Equation 8.3 is not identically* 0 *over the* $\{\boldsymbol{R}_k\}$ *matrices then* $p_x \bmod \mathcal{I}$ *is not identically* 0 *over the* $\{\boldsymbol{R}_k\}$ *matrices with overwhelming probability.*

Proof For every adversary $\mathcal{A}$ that queries a polynomial p for zero testing such that statement of the Claim is not true, we will construct an adversary $\mathcal{A}'$ that wins the game with probability 1 and does not issue such a zero testing query.[29] So it is enough to consider adversaries that does not make such zero testing queries.

Assume for the sake of contradiction that $p_x \bmod \mathcal{I}$ is identically 0 over $\{\boldsymbol{R}_k\}$ matrices. It means that there exists at least one coefficient in the monomial representation of $p_x \bmod \mathcal{I}$ (after substitution given in Equation 8.3) that is in the ideal $\mathcal{I}$ and is not equal to 0 in the ring R. We will construct an adversary $\mathcal{A}'$ that does not issue the zero testing query on the polynomial p and has a "winning strategy." $\mathcal{A}'$ (with non-uniform advice x) runs $\mathcal{A}$ until it queries the zero testing oracle on p. It then obtains p_x by setting all variables that are not in the matrices corresponding to evaluation of x to 0 and variables in the expansion of $\boldsymbol{\alpha}_x$ to 1 i.e it sets all variables that are not equal to $\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}, \widetilde{\boldsymbol{B}}_{\ell+1}$ to 0. It obtains ring handle h_1 after a zero testing query (It can obtain such an handle after evaluating the obfuscation on an input where the program's output is 0). $\mathcal{A}'$ constructs a new arithmetic circuit: q_x by substituting $\widetilde{\boldsymbol{B}}_0 \to \widetilde{\boldsymbol{A}}_0$, $\{\widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\} \to \{\widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}$ and $\widetilde{\boldsymbol{B}}_{\ell+1} \to \widetilde{\boldsymbol{A}}_{\ell+1}$ where

$$\widetilde{\boldsymbol{A}}_0 := \boldsymbol{A}_0 \times \boldsymbol{R}_0 \ ; \ \widetilde{\boldsymbol{A}}_{i,b_1,b_2} := \boldsymbol{R}^{adj}_{i-1} \times \boldsymbol{A}_{i,b_1,b_2} \times \boldsymbol{R}_i \ ; \ \widetilde{\boldsymbol{A}}_{\ell+1} := \boldsymbol{R}^{adj}_{\ell+1} \times \boldsymbol{A}_{\ell+1}$$

Note that q_x is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Let q'_x be equal to q_x with the restriction that all variables corresponding to $\boldsymbol{A}$ is set to 1. It is clear that q'_x is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. $\mathcal{A}'$ makes a post zeroizing query on q'_x with all variables (corresponding to $\{\boldsymbol{R}_k\}$ matrices) instantiated with the handle h_1. Since q'_x is not identically 0 and $q_x \bmod \mathcal{I}$ is equal to 0, q_x is valid winning post zeroizing query. ■

Claim 8.16 *If p_x is not identically 0 then $p_x \bmod \mathcal{I}$ is not identically 0.*

Proof The proof will be similar to that of Claim 8.15.

Assume for the sake of contradiction that $p_x \bmod \mathcal{I}$ is identically 0. It means that there exists at least one coefficient in the monomial representation of $p_x \bmod \mathcal{I}$ that is in the ideal $\mathcal{I}$ and is not equal to $0 \in R$. We will construct an adversary $\mathcal{A}'$ that can trivially "win" even without issuing a zero-testing query on the polynomial p. $\mathcal{A}'$ (with non-uniform advice x) runs $\mathcal{A}$ until it queries the zerotesting oracle on p. It then obtains p_x by setting all the variables that are not in the matrices corresponding to the evaluation of x to 0 i.e it sets all variables that are not equal to $\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}, \widetilde{\boldsymbol{B}}_{\ell+1}$ to 0. It obtains some ring handle h_1 after an appropriate zero testing query (by legitimately evaluating the program to 0). $\mathcal{A}'$ makes a post zeroizing query on p_x with all variables instantiated with the handle h_1. Since p_x is not identically 0 and $p_x \bmod \mathcal{I}$ is equal to 0, p_x is valid post zeroizing query. ■

Claim 8.17 *If p is not identically 0 then $p_{\mathcal{J}}$ is not identically 0.*

Proof The proof is identical to the proof of Claim 8.16 and hence omitted. ■

Claim 8.18 *Let $x \in D$ be such that the simulator outputs 0 in the iteration corresponding to x in the "for" loop in Figure 1. Then $\mathsf{Eval}(p_x) \bmod \mathcal{I} \neq 0$ with overwhelming probability.*

[29] By "winning" we imply that the adversary makes a post-zeroizing query that invokes the oracle $\mathcal{M}$ to reveal the entire secret state.

Proof We again have two cases to consider.

- $P(x) = 1$: In this case the simulator outputs 0 if the evaluation of $p_x \bmod \mathcal{I}$, that is Val_x on randomly chosen matrices $\widetilde{\boldsymbol{C}}_0, \{\widetilde{\boldsymbol{C}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}_{i\in[\ell]}, \widetilde{\boldsymbol{C}}_{\ell+1}$ from $R/\mathcal{I}$ is not 0. This in particular means that polynomial $p_x \bmod \mathcal{I}$ is not identically 0. In particular, p_x is not identically 0. By contrapositive of Item 1 in Corollary 8.8 and the definition of non-shortcutting matrices (c.f. Remark 5.3) we have that p_x is not identically 0 over the $\{\boldsymbol{R}_k\}$ matrices. From Claim 8.15 we have $p_x \bmod \mathcal{I}$ is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Note from the construction that each of the $\boldsymbol{R}_k$ matrices are sampled from $(D_{R,\sigma'})^{d\times d}$ where σ' is larger than the smoothing parameter of the lattice generated by $\mathcal{I}$. Hence, each entry of $\boldsymbol{R}_k \bmod \mathcal{I}$ is uniform in $R/\mathcal{I}$ (by Lemma 3.2). Therefore, $\mathsf{Eval}(p_x) \bmod \mathcal{I}$ is not 0 with overwhelming probability by Schwartz-Zippel lemma.
- $P(x) = 0$: We first claim that the simulator outputs 0 then the polynomial $p_x \bmod \mathcal{I}$ is not of the form $\gamma \cdot \widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{A}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{A}}_{\ell+1} \bmod \mathcal{I}$. This can be seen as follows: suppose $p_x \bmod \mathcal{I}$ is of the above form, then from the relationship between $\widetilde{\boldsymbol{C}}_0, \{\widetilde{\boldsymbol{C}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\}$ and $\widetilde{\boldsymbol{C}}_{\ell+1}$ (Equation 8.6) we infer that simulator's evaluation would yield 0. From the contrapositive of Claim 8.14 we infer p_x is not a constant multiple of the matrix product. Hence from Item 2 of Corollary 8.8 and the definition of non-shortcutting matrices (see Remark 5.3) we get p_x is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. From Claim 8.15 we have $p_x \bmod \mathcal{I}$ is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Now as in Case-1 of this proof we infer that $\mathsf{Eval}(p_x) \bmod \mathcal{I}$ is not 0 with overwhelming probability from Schwartz-Zippel lemma.

■

The following claim proves the "⇐" of Step 1 of Lemma 8.9.

Claim 8.19 *If* Sim *outputs* 0 *on zero testing on some polynomial* p *then with overwhelming probability* $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I}$ *is not 0.*

Proof Follows directly from Claims 8.18, 8.13. ■

We now prove Step 2 of Lemma 8.9.

Claim 8.20 *p is identically 0 if and only if* Sim *returns* $\mathsf{H}_{\mathsf{Rng}}(0)$ *with overwhelming probability.*

Proof "⇒": If p is identically 0 then $p_{\mathcal{J}}$ is identically 0. By algebraic independence of $\boldsymbol{\alpha}_x$ it follows that each of p_x for every $x \in D$ is identically 0. Hence, from Figure 1 we infer that Simulator outputs a ring handle to 0 in that case.

"⇐" If Sim returns $\mathsf{H}_{\mathsf{Rng}}(0)$ then it means that for every $x \in D$, evaluation of $p_x \bmod \mathcal{I}$ on random inputs in $R/\mathcal{I}$ is 0. Then by Schwartz-Zippel, we conclude that $p_x \bmod \mathcal{I}$ is identically 0 except with negligible probability for any arbitrary $x \in D$. Hence, by union bound (since $|D|$ is $\mathsf{poly}(\lambda)$) we conclude that for all $x \in D$, $p_x \bmod \mathcal{I}$ is identically 0 except with negligible probability. Therefore, for every $x \in D$, p_x is identically 0 with overwhelming probability (from Claim 8.16) and therefore $p_{\mathcal{J}}$ is identically 0 with overwhelming probability. Hence, p (from Claim 8.17) is identically 0 with overwhelming probability. ■

8.2.3 Correctness of the post-zeroizing queries in the simulation

We now prove that the simulator's responses are computationally indistinguishable to that of the oracle $\mathcal{M}$'s responses, that is, we argue that $\mathcal{M}$ also returns 0 on all the post zero testing queries with overwhelming probability.

Claim 8.21 *Consider the set of matrices $\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1}$ from the Eq. 8.3. Let*

$$p = c \cdot \widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i \in [\ell]} \widetilde{\boldsymbol{B}}_{i, y_{\mathsf{inp}_1(i)}, y_{\mathsf{inp}_2(i)}} \right) \times \widetilde{\boldsymbol{B}}_{\ell+1}$$

for some $c \in R$ and $y \in \{0,1\}^\lambda$ such that $P(y) = 0$. Then, p can be expressed as $c \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(y) + \boldsymbol{g} \cdot \boldsymbol{e}_y + \boldsymbol{k} \cdot \boldsymbol{g}^2)$ where $\boldsymbol{e}_y$ depends on the Kilian re-randomization and $\boldsymbol{k} \in R$.

Proof From the definition of $\widetilde{\boldsymbol{B}}$'s in Equation 8.3 we have

$$\begin{aligned} p &= c \cdot \widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i \in [\ell]} \widetilde{\boldsymbol{B}}_{i, y_{\mathsf{inp}_1(i)}, y_{\mathsf{inp}_2(i)}} \right) \times \widetilde{\boldsymbol{B}}_{\ell+1} \\ &= c \cdot \left(\widetilde{\boldsymbol{A}}_0 \times \left(\prod_{i \in [\ell]} \widetilde{\boldsymbol{A}}_{i, y_{\mathsf{inp}_1(i)}, y_{\mathsf{inp}_2(i)}} \right) \times \widetilde{\boldsymbol{A}}_{\ell+1} + \boldsymbol{k} \cdot \boldsymbol{g}^2 \right) \\ &= c \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(y) + \boldsymbol{g} \cdot \boldsymbol{e}_x + \boldsymbol{k} \cdot \boldsymbol{g}^2) \end{aligned}$$

■

Corollary 8.22 *Consider the set of matrices $\widetilde{\boldsymbol{B}}_0, \{\widetilde{\boldsymbol{B}}_{i,b_1,b_2}\}, \widetilde{\boldsymbol{B}}_{\ell+1}$ from the Eq. 8.3. Let*

$$p = c \cdot \widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i \in [\ell]} \widetilde{\boldsymbol{B}}_{i, y_{\mathsf{inp}_1(i)}, y_{\mathsf{inp}_2(i)}} \right) \times \widetilde{\boldsymbol{B}}_{\ell+1}$$

for some $c \in R$ and $y \in \{0,1\}^\lambda$ such that $P(y) = 0$. Then $p_{\mathcal{J}} = d \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(y) + \boldsymbol{g} \cdot \boldsymbol{e}_x)$ where $\boldsymbol{e}_y$ depends on the Kilian re-randomization.

The following claim proves that the ring handles returned by the oracle $\mathcal{M}$ are indistinguishable from uniform random elements in $R/\mathcal{I}$.

Claim 8.23 *If p is not identically 0 and $\mathsf{Eval}(p) \bmod \mathcal{I} = 0$, let $\mathsf{H}_{\mathsf{Rng}}(r_{\mathcal{M}})$ denote the handle returned by $\mathcal{M}$. Then,*

$$U_{R/\mathcal{I}} \overset{c}{\approx} r_{\mathcal{M}}$$

Proof First notice that since p is not identically 0, $p_{\mathcal{J}}$ is not identically 0 by Claim 8.17. Then, there exists at least one $x \in D$ where D refers to the single input decomposition of $p_{\mathcal{J}}$ such that p_x is not identically 0. In particular, $p_x \bmod \mathcal{I}$ (from Claim 8.16) is not identically 0. Since we are also given that $\mathsf{Eval}(p) \bmod \mathcal{I} = 0$, we conclude that $\mathsf{Eval}(p_{\mathcal{J}}) \bmod \mathcal{I} = 0$. Hence, from Claim 8.13 we infer that with overwhelming probability for each $x \in D$, $\mathsf{Eval}(p_x) \bmod \mathcal{I} = 0$. In

particular, $\mathsf{Eval}(p_{x^*}) \bmod \mathcal{I} = 0$. We first argue that with overwhelming probability $P(x^*) \neq 1$. If $P(x^*) = 1$ and since p_{x^*} is not identically 0, from Item 1 of Corollary 8.8 and Remark 5.3 we get p_{x^*} is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Hence from Claim 8.15 we get $p_{x^*} \bmod \mathcal{I}$ is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Since each entry of $\{\boldsymbol{R}_k\}$ matrix $\bmod \mathcal{I}$ is uniform (from Lemma 3.2) we get $\mathsf{Eval}(p_{x^*}) \bmod \mathcal{I}$ is not 0 with overwhelming probability from Schwartz-Zippel lemma. Hence we conclude that $P(x^*) = 0$. We now claim that p_{x^*} is a constant multiple of $\widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{B}}_{\ell+1}$. If it is not the case, then from Item 2 of Corollary 8.8 and Remark 5.3 we infer that p_{x^*} is not identically 0 over $\{\boldsymbol{R}_k\}$ matrices. Hence by a similar argument as made above we infer that with overwhelming probability that $\mathsf{Eval}(p_{x^*}) \bmod \mathcal{I}$ is not 0. We have p_{x^*} is a constant multiple of $\widetilde{\boldsymbol{B}}_0 \times \left(\prod_{i\in[\ell]} \widetilde{\boldsymbol{B}}_{i,x_{\mathsf{inp}_1(i)},x_{\mathsf{inp}_2(i)}}\right) \widetilde{\boldsymbol{B}}_{\ell+1}$. From Corollary 8.22 we can deduce that $p_{x^*} = c \cdot (\boldsymbol{g} \cdot \boldsymbol{Q}^\chi(x^*) + \boldsymbol{g} \cdot \boldsymbol{e}_{x^*})$. Hence, $\frac{p_{x^*}}{\boldsymbol{g}} \bmod \mathcal{I} = \boldsymbol{Q}^\chi(x^*) + \boldsymbol{e}_{x^*} \bmod \mathcal{I}$.

Let us define the randomized procedure $\mathsf{samp}(\gamma; r)$ that samples uniformly $[-\gamma/2+1, \gamma/2]^n$ using r as random coins. Notice that $\boldsymbol{Q}^\chi(y)$ from our construction is equivalent to $\mathsf{samp}(\gamma; \mathcal{PRF}_\chi(y))$ where $\mathcal{PRF}_\chi(y) = (\mathcal{PRF}_{\psi_1}(y), \cdots, \mathcal{PRF}_\psi(y))$. Hence, $Z = \boldsymbol{Q}^\chi(y) \bmod \mathcal{I} = \mathsf{samp}(\gamma; \mathcal{PRF}_\chi(y)) \bmod \mathcal{I}$. From the security of $\mathcal{PRF}$ we observe that Z is computationally indistinguishable from Z' where $Z' = \mathsf{samp}(\gamma; r)$ where r is chosen uniformly at random. Observe from Lemma 7.4 that Z' is statistically close to uniform distribution in $R/\mathcal{I}$.

Therefore, $\boldsymbol{Q}^\chi(x^*) + \boldsymbol{e}_{x^*} \bmod \mathcal{I}$ is computationally indistinguishable from $R/\mathcal{I}$. ■

Finally, note that from Claim 8.23 $\mathcal{M}$ returns a ring handle to a non zero element then the distribution of the element is computationally indistinguishable from uniform distribution in $R/\mathcal{I}$. Hence, any non-zero polynomial (with $\mathsf{poly}(\lambda)$ degree) on such ring elements evaluate to 0 with negligible probability from Schwartz-Zippel lemma which implies that only with negligible probability the post-zeroizing query will be successful. Hence, with overwhelming probability the oracle $\mathcal{M}$ would return 0 for the post-zeroizing queries.

References

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 528–556, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[AGHS12] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Sampling discrete gaussians efficiently and obliviously. Cryptology ePrint Archive, Report 2012/714, 2012. http://eprint.iacr.org/.

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal obfuscation and witness encryption: Boosting correctness and combining security. Cryptology ePrint Archive, Report 2016/281, 2016. http://eprint.iacr.org/.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in np cap conp. *J. ACM*, 52(5):749–765, 2005.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5, 1986.

[BF11] Dan Boneh and David Mandell Freeman. Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 1–16, Taormina, Italy, March 6–9, 2011. Springer, Heidelberg, Germany.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 221–238, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. *EUROCRYPT 2016*, 2016:167, 2016.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 1–25, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[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 Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 3–12, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[CLLT15] Jean-Sebastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of GGH15 multilinear maps. Cryptology ePrint Archive, Report 2015/1037, 2015. http://eprint.iacr.org/2015/1037.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 267–286, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[DN12] Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 433–450, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany.

[Gar15] Sanjam Garg. *Candidate Multilinear Maps*. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA, 2015.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[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 Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 498–527, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

[GLSW15] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 151–170, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

[GS02] Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 299–320, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany.

[HJ15] Yupu Hu and Huiwen Jia. Cryptanalysis of GGH map. Cryptology ePrint Archive, Report 2015/301, 2015. http://eprint.iacr.org/2015/301.

[Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. Cryptology ePrint Archive, Report 2016/257, 2016. http://eprint.iacr.org/.

[LSS14] Adeline Langlois, Damien Stehlé, and Ron Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Computing*, 37(1):267–302, 2007.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. Cryptology ePrint Archive, Report 2016/147, 2016. http://eprint.iacr.org/.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 500–517, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[SS11] Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 27–47, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 439–467, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.