Indistinguishability Obfuscation from Semantically-Secure Multilinear Encodings

Rafael Pass[*] Karn Seth[†] Sidharth Telang[‡]

March 12, 2014

Abstract

We define a notion of semantic security of multilinear (a.k.a. graded) encoding schemes, which generalizes a multilinear DDH assumption: roughly speaking, we require that if two constant-length sequences $\vec{m}_0$, $\vec{m}_1$ are *pointwise statistically indistinguishable* by algebraic attackers C (obeying the multilinear restrictions) in the presence of some other elements $\vec{z}$, then encodings of these sequences should be computationally indistinguishable. Assuming the existence of semantically secure multilinear encodings and the LWE assumption, we demonstrate the existence of indistinguishability obfuscators for all polynomial-size circuits.

We rely on the beautiful candidate obfuscation constructions of Garg et al (FOCS'13), Brakerski and Rothblum (TCC'14) and Barak et al (EuroCrypt'14) that were proven secure only in idealized generic multilinear encoding models, and develop new techniques for demonstrating security in the standard model, based only on semantic security of multilinear encodings (which trivially holds in the generic multilinear encoding model).

[*]Cornell University, Cornell NYC Tech. Email: `rafael@cs.cornell.edu`. Work supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

[†]Cornell University. Email: `karn@cs.cornell.edu`.

[‡]Cornell University. Email: `sidtelang@cs.cornell.edu`.

1 Introduction

The goal of *program obfuscation* is to "scramble" a computer program, hiding its implementation details (making it hard to "reverse-engineer"), while preserving the functionality (i.e, input/output behavior) of the program. Precisely defining what it means to "scramble" a program is non-trivial: on the one hand, we want a definition that can be plausibly satisfied, on the other hand, we want a definition that is useful for applications.

A first formal definition of such program obfuscation was provided by Hada [Had00]: roughly speaking, Hada's definition—let us refer to it as *strongly virtual black-box*—is formalized using the simulation paradigm. It requires that anything an attacker can learn from the obfuscated code, could be simulated using just black-box access to the functionality.[1] Unfortunately, as noted by Hada, only learnable functionalities can satisfy such a strong notion of obfuscation: if the attacker simply outputs the code it is given, the simulator must be able to recover the code by simply querying the functionality and thus the functionality must be learnable.

An in-depth study of program obfuscation was initiated in the seminal work of Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI+01]. Their central result shows that even if we consider a more relaxed simulation-based definition of program obfuscation—called *virtual black-box (VBB)* obfuscation—where the attacker is restricted to simply outputting a single bit, impossibility can still be established (assuming the existence of one-way functions).[2] Their result is even stronger, demonstrating the existence of families of functions such that given black-box access to f_s (for a randomly chosen s), not even a *single* bit of s can be guessed with probability significantly better than 1/2, but given the code of any program that computes f_s, the entire secret s can be recovered. Thus, even quite weak simulation-based notions of obfuscation are impossible.

But weaker notions of obfuscation may be achievable, and may still suffice for (some) applications. Indeed, Barak *et al.* [BGI+01] also suggested two such notions:

- The notion of *indistinguishability obfuscation*, first defined by Barak *et al.* [BGI+01] and explored by Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b], roughly speaking requires that obfuscations $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ of any two *equivalent* circuits C_1 and C_2 (i.e., whose outputs agree on all inputs) from some class $\mathcal{C}$ are computationally indistinguishable.

- The notion of *differing-input obfuscation*, first defined by Barak *et al.* [BGI+01] and explored by Boyle, Chung and Pass [BCP14] and by Ananth, Boneh, Garg, Sahai and Zhandry [ABG+13] strengthens the notion of indistinguishability obfuscation to also require that even if C_1 and C_2 are not equivalent circuits, if an attacker can distinguish obfuscations $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$, then the attacker must "know" an input x such that $C_1(x) \neq C_2(x)$, and this input can be efficiently "extracted" from A.

In a very recent breakthrough result, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH+13b] provided the first candidate constructions of indistinguishability obfuscators for all polynomial-size circuits so-called, based on so-called *multilinear (a.k.a. graded) encodings* [BS03, Rot13, GGH13a]—for which candidate constructions were recently discovered in the seminal work of Garg, Gentry and Halevi [GGH13a], and more recently, alternative constructions were provided by Coron, Lepoint and Tibouchi [CLT+13].

The obfuscator construction of Garg et al proceeds in two steps. They first provide a candidate construction of an indistinguishability obfuscator for NC^1 (this construction is essentially assumed to be secure); next, they demonstrate a "bootstrapping" theorem showing how to use fully homomorphic

[1]Hada actually considered a slight distributional weakening of this definition.

[2]A similar notion of security (without referring to obfuscation) was considered even earlier by Canetti [Can97] in the special case of what is now referred to as *point-function obfuscation*.

encryption (FHE) schemes [Gen09] and indistinguishability obfuscators for NC^1 to obtain indistinguishability obfuscators for all polynomial-size circuits.

Further constructions of obfuscators for NC^1 were subsequently provided by Brakerski and Rothblum [BR14] and Barak, Garg, Kalai, Paneth and Sahai [BGK+13]—in fact, these constructions achieve even stronger notions of virtual-black-box obfuscation in idealized "generic" multilinear encoding models. Additionally, Boyle, Chung and Pass [BCP14] present an alternative bootstrapping theorem, showing how to employ on extractability obfuscation for NC^1 to obtain differing-input (and thus also indistinguishability) obfuscation for both circuits and Turing machines. (Ananth et al [ABG+13] also provide Turing machine differing-input obfuscators, but start instead from differing-input obfuscators for polynomial-size circuits.)

In parallel with the development of candidate obfuscation constructions, several surprising applications of both indistinguishability and extractability obfuscations have emerged: for instance, in the works of Garg et al [GGH+13b], Sahai and Waters [SW13], Hohenberger, Sahai and Waters [HSW13], Boyle, Chung and Pass [BCP14], Boneh and Zhandry [BZ13], Garg, Gentry, Halevi and Raykova [GGHR14], Bitansky, Canetti, Paneth and Rosen [BCPR13], Boyle and Pass [BP13]. Most notable among these is the work of Sahai and Waters [SW13] (and the "punctured program" paradigm it introduces) which shows that for some interesting applications of virtual black-box obfuscation (such as turning private-key primitives into public-key one), the weaker notion of indistinguishability obfuscation suffices. Additionally, as shown by Goldwasser and Rothblum [GR07], indistinguishability obfuscators provide a very nice "best-possible" obfuscation guarantee: if a functionality can be VBB obfuscated (even non-efficiently!), then any indistinguishability obfuscator for this functionality is VBB secure.

1.1 Towards "Provably-Secure" Obfuscation

But despite these amazing developments, the following question remains wide open:

> *Can the security of general-purpose obfuscations be* **reduced** *to some "natural" intractability assumption?*

Note that while the construction of indistinguishability obfuscation of Garg et al is based on *some* intractability assumption, the assumption is very tightly tied to their scheme—in essence, the assumption stipulates that the scheme is a secure indistinguishability obfuscator. Rather, we are here concerned with the question of whether some *general* assumption (that is interesting in its own right, and is not "tailored" to the scheme) can be used to obtain indistinguishability obfuscation. More importantly, we are interested in *reducing* the security of the obfuscation to some *simpler* assumption—that is, we are not interested in assumptions that "directly" (without any security reduction that deals with *arbitrary* nuPPT attackers) imply security of the obfuscation.

The VBB constructions of Brakerski and Rothblum [BR14] and Barak et al [BGK+13] give us more confidence in the plausible security of their obfuscators, in that they show that at least "generic" attacks—that treat multilinear encoding as if they were "physical envelopes" on which multilinear operations can be performed—cannot be used to break security of the obfuscators. But at the same time, non-generic attacks against their scheme are known—since general-purpose VBB obfuscation is impossible! Thus, it is not clear to what extent security arguments in the generic multilinear encoding model should make us more confident that these constructions satisfy e.g., a notion of indistinguishability obfuscation. (We mention that the question of to what extent one can capture such "real-world" security properties from security proofs in the generic model through a "meta-assumption" was raised, but not investigated, in [BGK+13]; see Remark 1.) In particular, one way to interpret the results of [BR14, BGK+13] is that the assumption *"any scheme that is secure in the generic multilinear encoding model is secure in the standard model"* is false; not only is it false, but it fails for a natural scheme.

In light of this, a natural weakening of this assumption may instead be *"any indistinguishability property that holds in the generic generic multilinear encoding model also holds in the the standard* *model"*; if we only care about obtaining indistinguishability obfuscation, such an assumption would also suffice. Looking ahead, as we show in Theorem 3, such an assumption is false. Furthermore, even if this assumption were to be true[3], it would not help us in addressing the above question since the assumption *directly* implies indistinguishability security of obfuscations (that are secure against generic attacks), without any security reduction. Of course, there is a security proof required to prove security in the generic encoding model, but it only deals with "idealized" generic attackers; for such attackers, the security reduction does not have to "commit" to the code it outputs but can instead adaptively "program" answers to "algebraic queries" based on what the queries are. Rather, what we are interested in is a security argument that applies to *arbitrary nuPPT* attackers; what makes this particularly challenging is that now the security reduction must output a fully specified code before it knows how the attacker plans to use it.

In this work, we address the above question. We stipulate a new, but in our eyes natural, assumption regarding multilinear encodings—the existence of, so-called, *semantically-secure multilinear encodings*, and show how to construct indistinguishability obfuscators for NC^1 (which then can be bootstrapped up to general circuits) based on this assumption, using a non-trivial security reduction—looking ahead, we show how to demonstrate indistinguishability obfuscation by relying on a hybrid argument that enables us to transition from the obfuscation of one program to another; we rely on our assumption to prove indistinguishability of each of the consecutive hybrids. As we shall explain shortly, this assumption can be viewed as a "generalized DDH" or "uber-assumption", as in [BBG05], in the context of multilinear maps.[4]

1.2 Obfuscation From Semantically-secure Multilinear Encodings

Semantically-secure multilinear encodings (or The "Generalized DDH Assumption") Recall that a multilinear (a.k.a. graded) encoding scheme [GGH13a, GGH+13b] enables anyone that has access to a *public parameter* pp and *encodings* $E_S^x = \mathsf{Enc}(x, S)$, $E_S^y = \mathsf{Enc}(y, S')$ of ring elements x, y under the sets $S, S' \subset [n]$ to *efficiently*:[5]

- compute an encoding $E_{S \cup S'}^{x \cdot y}$ of $x \cdot y$ under the set $S \cup S'$, as long as $S \cap S' = \emptyset$;
- compute an encoding E_S^{x+y} of $x + y$ under the set S as long as $S = S'$;
- compute an encoding E_S^{x-y} of $x - y$ under the set S as long as $S = S'$.

(Given just access to the public-parameter pp, generating an encoding to a particular element x may not be efficient; however, it can be efficiently done given access to the *secret parameter* sp.) Additionally, given an encoding E_S^x where the set S is the whole universe $[n]$—called the "target set"—we can efficiently check whether $x = 0$ (i.e., we can "zero-test" encodings under the target set $[n]$.) In essence, multilinear encodings enable computations of certain restricted set (determined by the sets S under which the elements are encoded) of arithmetic circuits, and finally determine whether the output of the circuit is 0.

Towards explaining our notion of semantical security, let us first consider a DDH-like assumption for multilinear encoding (similar in spirit to the "graded DDH" assumption of Garg et al [GGH13a] for

[3]A reasonable variant of it, which we refer to as the "Uber-Uber Assumption" is to demand that "any indistinguishability property that holds in a *statistical sense* in the generic encoding model also holds *computationally* in the standard model". See Section 1.2 for more details.

[4]We thank Shai Halevi for pointing out the connection with [BBG05].

[5]Just as [BR14, BGK+13], we here rely on "set-based" graded encoding; these were originally called "generalized" graded encodings in [GGH13a]. Following [GGH+13b, BGK+13] (and in particular the notion of a "multilinear jigsaw puzzles" in [GGH+13b]), we additionally enable anyone with the secret parameter to encode *any* elements (as opposed to just *random* elements as in [GGH13a]).

"symmetric" multilinear encodings). Consider sampling n random elements $\vec{z}$, and let m_0 be the product of the elements in $\vec{z}$, and m_1 be just a random element. A DDH-like assumption would require that encodings of m_0 and m_1 under the "target" set S are indistinguishable, given encodings of $\vec{z}$ under sets $\vec{T}$, if S is not the disjoint union of the sets in $\vec{T}$; that is, distinguishing an encoding of the product of all elements in $\vec{z}$ is indistinguishable from the encoding of a uniform element, as long as the encodings are made under sets that prohibit "legally" obtaining the product of the elements.

We here consider a "generalized DDH-type" assumption in the multilinear setting, similar in spirit to the "uber-assumption" of [BBG05] (which was considered in the context of bilinear maps), where we not only care about products of all the elements in $\vec{z}$ but also more complicated relations among the elements and sets they are encoded under: we consider elements $\vec{z}$, m_0, m_1 that come from *any* efficient distribution D that (with overwhelming probability) makes it impossible to tell apart m_0 and m_1 in the presence of $\vec{z}$ using "legal" algebraic operations respecting the sets (but otherwise being *computationally unbounded*).

More precisely, our notion of *single-message* semantic security for multilinear encodings requires that for every $S, \vec{T}$ pair, every "valid" distribution D over $m_0, m_1, \vec{z}$—where a distribution D is valid if for every (even computationally unbounded) algebraic attacker (obeying the set-restrictions)[6], with overwhelming probability over $m_0, m_1, \vec{z} \leftarrow D$, A produces identically the same output when given access to $(m_0, \vec{z})$ or $(m_1, \vec{z})$: that is, the elements are *pointwise statistically* indistinguishable by legal algebraic attackers—we have that encodings $\{\mathsf{Enc}(m_0, S), \mathsf{Enc}(\vec{z}, \vec{T})\}$ and $\{\mathsf{Enc}(m_1, S), \mathsf{Enc}(\vec{z}, \vec{T})\}$ are computationally indistinguishable (by standard nuPPT attackers). In other words, encodings of $m_0, \vec{z}$ and $m_1, \vec{z}$ are computationally indistinguishable if, with overwhelming probability, legal algebraic attackers operate *identically the same* on $(m_0, \vec{z})$ and on $(m_1, \vec{z})$.

In our setting, we will require a slight strengthening of the above notion to a *constant*-message setting, where m_0, m_1, and S are replaced by *constant-length* vectors $\vec{m_0}, \vec{m_1}, \vec{S}$. In the remainder of the paper, we simply refer to constant-message semantically-secure encodings as *semantically-secure multilinear encodings.* Note that (randomized) multilinear encoding schemes in the generic multilinear encoding model of [BGK$^+$13] are trivially semantically secure. In essence, the assumption of semantic security, stipulates that for the *particular task* of distinguishing encodings of a *constant* number of elements (computationally unbounded) generic attacks cannot be (significantly) beaten.

Let us point out that one may consider a "multi-message" notion of semantical security (or an "uber-uber generalized DDH" assumption), which considers *polynomial-length* (as opposed to constant-length) sequences of messages $\vec{m_0}$, $\vec{m_1}$. Such a notion essentially requires that any indistinguishability property that holds in a *statistical pointwise* sense against generic attackers, also holds *computationally* against a nuPPT attacker that sees the actual encodings. The constructions of [BR14, BGK$^+$13] indeed satisfy indistinguishability obfuscation in a statistical pointwise sense (w.r.t. to generic attackers)[7] and thus it "directly" follows that these constructions satisfy indistinguishability obfuscation under the multi-message semantical security assumption.[8] Whereas we are not aware of any attacks against such an multi-message semantical security assumption, our focus here is on an assumption that is as close as possible to a "DDH-type" assumption (and consequently as weak as possible), and thus we focus on indistinguishability of only a *constant* number of elements. Furthermore, as mentioned above, towards the goal of *reducing* the security of obfuscation to some assumption, we are also less interested in assumptions that "directly" imply security of the scheme, without any type of security reduction w.r.t.,

[6]Even more precisely, we allow the attacker to be computationally unbounded while making polynomially many (or even subexponentially many) algebraic zero-test queries.

[7]This follows directly from the proofs in [BR14, BGK$^+$13] although it wasn't explicitly noted.

[8]In fact, any obfuscation that 1) only releases encodings of elements, and 2) satisfies *statistical pointwise* indistinguishability obfuscation against generic attackers, as the schemes of [BR14, BGK$^+$13] do, is secure by definition under this "uber-uber" assumption. We thank Sanjam Garg for pointing this out.

nuPPT attackers.[9]

Obfuscation from Semantically Secure Multilinear Encodings Our central result shows how to construct indistinguishability obfuscators for NC^1 based on the existence of (constant-message) semantically-secure multilinear encodings.

Theorem 1 (Informally stated)**.** *Assume the existence of semantically secure multilinear encodings. Then there exists indistinguishability obfuscators for* NC^1*.*

As far as we know, this is the first result presenting indistinguishability obfuscators for NC^1 based on any type of assumption with a "non-trivial" security reduction w.r.t. arbitrary nuPPT attackers (as opposed to restricted "generic" attackers).

If additionally assuming the existence of a leveled FHE [RAD78, Gen09] with decryption in NC^1—implied, for instance, by the LWE assumption [BV11, BGV12]—this construction can be bootstrapped up to obtain indistinguishability obfuscators for all polynomial-size circuits by relying on the technique from [GGH+13b].

Theorem 2 (Informally stated)**.** *Assume the existence of semantically secure multilinear encodings and a leveled FHE with decryption in* NC^1*. Then there exists indistinguishability obfuscators for P/poly.*

Semantical Security w.r.t. Restricted Classes of Distributions Our most basic notion of semantical security requires indistinguishability to hold w.r.t. to *any* "valid" message distribution. This may seem like a strong assumption. Firstly, such a notion can clearly not be satisfied by a *deterministic* encoding schemes (as envisioned in the original work of [BS03])—we can never expect encodings of 0 and 1 (under a non target set, and without any auxiliary inputs) to be indistinguishable. Secondly, even if we have a randomized encoding scheme in mind (such as the candidates of [GGH13a, CLT+13]), giving the attacker access to encodings of *arbitrary* elements may be dangerous: As mentioned in [GGH13a], attacks (referred to as "weak discrete logarithm attacks") on their scheme are known in settings where the attacker can get access to "non-trivial" encodings of 0 under any *non-target* set $S \subset [k]$. (We mention that, as far as we know, no such attacks are currently known on the candidate construction of [CLT+13].)

For the purposes of the results in our paper, however, it suffices to consider a notion of semantical security w.r.t. *restricted classes of distributions* D. In particular, to deal with both of the above issues, we consider "high-entropy" distributions D that sample elements $\vec{m}_0, \vec{m}_1, \vec{z}$ such that 1) each individual element has high-entropy, and 2) any element, associated with a *non-target* set $S \subset [k]$, that can be obtained by applying "legal" algebraic operations to $(\vec{m}_b, \vec{z})$ (for $b \in \{0,1\}$) has high-entropy (and thus is non-zero with overwhelming probability).[10] We refer to such a notion as *entropic semantical security.*

Additionally, for our purposes, it suffices to restrict to a setting where e.g., the number of encodings released is $O(k)$[11], and where the sets $\vec{S}, \vec{T}$ are pairs of indices $\{i, j\}$, $i, j \in [k]$.

On Falsifiability of Semantically Secure Multilinear Encodings Let us point out that the assumption that a multilinear encoding scheme is (entropic) semantically secure is not necessarily "efficiently falsifiable" in the terminology of Naor [Nao03], since checking whether there exists some legal algebraic attacker telling apart two constant-length sequences of elements (in the presence of some other elements z) is not necessarily polynomial-time computable. Note, however, that the assumption that a particular scheme is an indistinguishability obfuscator is not an efficiently falsifiable assumption either: a presumed attacker must exhibit two functionally-equivalent circuits C_1 and C_2 that it can

[9] As mentioned above, while the works of [BR14, BGK+13] do provide security proofs, these proofs only consider generic attackers; our focus is on security reductions that apply to arbitrary nuPPT attackers.

[10] Technically, by high-entropy, we here mean that the min-entropy is at least $\log|R| - O(\log\log|R|)$ where R is the ring associated with the encodings; that is, the min-entropy is "almost" optimal (i.e., $\log|R|$).

[11] We thank Hoeteck Wee for encouraging us to make this explicit.

distinguish obfuscations of; but checking whether two circuits are functionally equivalent may not be polynomial-time computable.[12]

On the other hand, for many *applications* of indistinguishability obfuscation (e.g., functional encryption [GGH$^+$13b]) it suffices to require indistinguishability for restricted classes of programs that (with overwhelming probability) are functionally equivalent, and for which it can be *efficiently checked* that the distribution over programs belong to the class (and thus the two circuits selected indeed are functionally equivalent).[13] For such a "class-specific" notion of obfuscation, it suffices for us to consider restricted classes of distributions for which semantical security now becomes falsifiable.

More generally, such applications of indistinguishability obfuscation can be based on a "meta-falsfiable" variant of (entropic) semantical security where the attacker not only needs to come up with a valid message distribution (for which it can distinguish encodings) but also a *mathematical proof* (in some appropriate language) that the distribution is valid.

We leave open the intruiging question of whether *general-purpose* (as opposed to application-specific) indistinguishability obfuscation can be based on some falsifiable assumption; the lower-bound from [GGSW13] regarding witness encryption (which is implied by indistinguishability obfuscation), provides some indications of why basing indistinguishability obfuscation on a falsifiable assumption may be hard.

1.3 Construction Overview

Following the original work of Garg et al (as well as subsequent works), our construction proceeds in three steps:

- We view the NC^1 circuit to be obfuscated as a *branching program* BP (using Barrington's Theorem [Bar86])—that is, the program is described by m pairs of matrices $(B_{i,0}, B_{i,1})$, each one labelled with an input bit $\mathsf{inp}(i)$, and the program is evaluated computing by for each $i \in [m]$, choosing one of the two matrices $(B_{i,0}, B_{i,1})$, based on the input, computing the product, and finally based on the product determining the output—there is a unique "accept" (i.e., output 1) matrix, and a unique "reject" (i.e., output 0) matrix.

- The branching program BP is *randomized* using Kilian's technique [Kil88] (roughly, each pair of matrices is appropriately multiplied with the same random matrix R while ensuring that the output is the same), and then "randomized" some more—each individual matrix is multiplied by a random *scalar* α. Let us refer to this step as Rand.

- Finally the randomized matrices are encoded using multilinear encodings with the sets selected appropriately. We here rely on a (simple version) of the *straddling set* idea of [BGK$^+$13] to determine the sets.[14] We refer to this step as Encode.

(The original construction as well as the subsequent works also consisted of several other steps, but for our purposes these will not be needed.) The obfuscated program is now evaluated by using the multilinear operations to evaluate the branching program and finally appropriately use the zero-test to determine the output of the program. Let us refer to this construction as the "basic obfuscator".

Roughly speaking, the idea behind the basic obfuscator is that the multilinear encodings *intuitively* ensure that any distinguisher (attacker) getting the encoding needs to multiply matrices along paths that corresponds to some input to the branching program (the straddling sets are used to ensure that

[12]In fact, assuming the existence of indistinguishability obfuscation and one-way functions it is easy to come up with a method to sample C_1 and C_2 that with high probability compute different functions, yet are indistinguishable; see the lower bound for witness encryption of [GGSW13].

[13]A notable exception is the construction of witness encryption from indistinguishability obfuscation of [GGH$^+$13b].

[14]Although we have not verified all the details, it seems that we could have also relied on the simpler encoding method from [GGH$^+$13b, BR14] where each matrix is encoded with different *singleton set*, but using straddling sets somewhat simplifies the analysis.

the input is used consistently in the evaluation)[15]; the scalars α ensure that a potential distinguisher without loss of generality can use a *single* "multiplication-path" and still succeed with roughly the same probability, and finally, Kilian's randomization steps ensures that if a distinguisher *only* operates on matrices along a single path that corresponds to some input x (in a consistent way), then its output can be perfectly simulated given just the output of the circuit on input x. (The final step relies on the fact that the output of the circuit uniquely determines product of the branching program along the path, and Kilian's randomization then ensures that the matrices along the path are random conditioned on the product being this unique value.) Thus, if a distinguisher can tell apart obfuscations of two programs BP_0, BP_1, there must exist some input on which they produce different outputs. The above intuitions can indeed be formalized w.r.t. *generic attackers* (that only operate on the encodings in a legal way, respecting the set restrictions), relying on arguments from [BR14, BGK$^+$13]. However, although security w.r.t. generic attackers will be useful to us (as we shall see shortly), we are interested in proving security w.r.t. *all polynomial-size attackers*.

Towards this, we will add an additional program transformation steps before the Rand and Encode steps. Roughly speaking, we would like to have a method $\mathsf{Merge}(BP_0, BP_1, b)$ that "merges" BP_0 and BP_1 into a single branching program that evaluates BP_b; additionally, we require that $\mathsf{Merge}(BP_0, BP_1, 0)$ and $\mathsf{Merge}(BP_0, BP_1, 1)$ only differ in a constant number of matrices. We achieve this merge procedure by connecting together BP_0, BP_1 into a branching program of double width and adding two "switch" matrices in the beginning and the end, determining if we should go "up" or "down". Thus, to switch between $\mathsf{Merge}(BP_0, BP_1, 0)$ (which is functionally equivalent to BP_0) and $\mathsf{Merge}(BP_0, BP_1, 1)$ (which is functionally equivalent to BP_1) we just need to switch the "switch matrices". More precisely, given branching programs BP_0 and BP_1 described respectively by pairs of matrices $\{(B^0_{i,0}, B^0_{i,1}),\ (B^1_{i,0}, B^1_{i,1})\}_{i\in[m]}$, we construct a merged program $\mathsf{Merge}(BP_0, BP_1, b)$ described by $\{(\hat{B}^0_{i,0}, \hat{B}^0_{i,1})\}_{i\in[m]}$ such that

$$\hat{B}^0_{i,b} = \hat{B}^1_{i,b} = \begin{pmatrix} B^0_{(i-1),b} & 0 \\ 0 & B^1_{(i-1),b} \end{pmatrix} \quad \text{for all } 2 \le i \le m+1 \text{ and } b \in \{0,1\}$$

and the first and last matrices are given by:

$$\hat{B}^0_{1,b} = \hat{B}^0_{m+2,b} = I_{2w\times 2w} \qquad \text{for } b \in \{0,1\}$$
$$\hat{B}^1_{1,b} = \hat{B}^1_{m+2,b} = \begin{pmatrix} 0 & I_{w\times w} \\ I_{w\times w} & 0 \end{pmatrix} \qquad \text{for } b \in \{0,1\}$$

It directly follows from the construction that $\mathsf{Merge}(BP_0, BP_1, 0)$ and $\mathsf{Merge}(BP_0, BP_1, 1)$ differ only in the first and the last matrices (i.e., the "switch" matrices). Furthermore, it is not hard to see that $\mathsf{Merge}(BP_0, BP_1, b)$ is functionally equivalent to BP_b.

Our candidate obfuscator is now defined as $i\mathcal{O}(B) = \mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP, I, 0)))$, where I is simply a "dummy" program of the same size as BP.[16]

The idea behind the merge procedure is that to prove that obfuscations of two programs BP_0, BP_1 are indistinguishable, we can come up with a sequence of hybrid experiments that start with $i\mathcal{O}(BP_0)$ and end with $i\mathcal{O}(BP_1)$, but between any two hybrids only changes a constant number of encodings, and thus intuitively we may rely on semantic security of multilinear encodings to formalize the above intuitions. At a high level, our strategy will be to matrix-by-matrix, replace the dummy branching program in the obfuscation of BP_0 with the branching program for BP_1. Once the entire dummy branching program has been replaced by BP_1, we flip the "switch" so that the composite branching program now computes the branching program for BP_1. We then replace the branching program for

[15]The encodings, however, still permit an attacker to add elements within matrices.

[16]This description oversimplifies a bit. Formally, the Rand step needs to depends on the field size used in the Encode steps, and thus in our formal treatment we combine these two steps together.

BP_0 with BP_1, matrix by matrix, so that we have two copies of the branching program for BP_1. We now flip the "switch" again, and finally restore the dummy branching program, so that we end up with one copy of BP_1 and one copy of the dummy, which is now a valid obfuscation of BP_1. In this way, we transition from an obfuscation of BP_0 to an obfuscation of BP_1, while only changing a small piece of the obfuscation in each step. (On a very high-level, this approach is somewhat reminiscient of the Naor-Yung "two-key" approach in the context of CCA security [NY90] and the "two-key" bootstrapping result for indistinguishability obfuscation due to Garg et al [GGH+13b]—in all these approaches the length of the scheme is artificially doubled to facilitate a hybrid argument. It is perhaps even more reminiscient of the Feige-Shamir "trapdoor witness" approach for constructing zero-knowledge arguments [FS90], whereby an additional "dummy" trapdoor witness is introduced in the construction to enable the security proof.)

More precisely, consider the following sequence of hybrids.

- We start off with $i\mathcal{O}(BP_0) = \mathsf{Enc}(\mathsf{Rand}(\mathsf{Merge}(BP_0, I, 0)))$
- We consider a sequence of hybrids where we gradually change the dummy program I to become BP_1; that is, we consider $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_0, BP', 0)))$, where BP' is "step-wise" being populated with elements from BP_1.
- We reach $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_0, BP_1, 0)))$.
- We turn the "switch" : $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_0, BP_1, 1)))$.
- We consider a sequence of hybrids where we gradually change the BP_0 to become BP_1; that is, we consider $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP', BP_1, 1)))$, where BP' is "step-wise" being populated with elements from BP_1.
- We reach $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_1, BP_1, 1)))$.
- We turn the "switch" back: $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_1, BP_1, 0)))$.
- We consider a sequence of hybrids where we gradually change the second BP_1 to become I; that is, we consider $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_1, BP', 0)))$, where BP' is "step-wise" being populated with elements from I.
- We reach $\mathsf{Encode}(\mathsf{Rand}(\mathsf{Merge}(BP_1, I, 0))) = i\mathcal{O}(BP_1)$.

By construction we have that if BP_0 and BP_1 are functionally equivalent, then so will all the hybrid programs–the key point is that we only "morph" between two branching programs on the "inactive" part of the merged branching program. Furthermore, by construction, between any two hybrids we only change a constant number of elements. Thus, if some distinguisher can tell apart $i\mathcal{O}(BP_0)$ and $i\mathcal{O}(BP_1)$, it must be able to tell apart two consecutive hybrids. But, by semantic security it then follows that some "legal" algebraic attacker can tell apart the encodings in the two hybrids. Roughly speaking, we can now rely on indistinguishable security of the basic obfuscator w.r.t. to just *generic* attackers to complete the argument.

There is a catch with the final step though. Recall that to rely on Kilian's simulation argument it was crucial that there are unique accept and reject matrices. For our "merged" programs, this is no longer the case (the output matrix is also a function of the second "dummy" program). We overcome this issue by noting that the *first column* of the output matrix actually is unique, and this is all we need to determine the output of the branching program. Consequently it suffices to release encodings of the *just* first column (as opposed to the whole matrices) of the last matrix pair in the branching program, and we can still determine the output of the branching program. As we show, for such a modified scheme, we can also simulate the (randomized) matrices along an "input-path" given just the first column of the output matrix. This concludes the description of our indistinguishability obfuscator.

1.4 On the Impossibility of Extractable Security

A natural question is whether there are reasonable qualitative strengthenings of semantical security that can be used to achieve stronger notions of obfuscation, such as differing-input (a.k.a. extractability) obfuscation. We here consider such a strengthening: roughly speaking, *extractable semantic security* (or the "extractable uber assumption") of multilinear encodings strengthens the notion of semantic security by requiring that if an attacker A can distinguish between encodings of a constant number of elements (in the presence of auxiliary encodings), then there exists an *efficient* "algebraic" strategy that distinguishes the elements; that is, the algebraic operations needed to distinguish the elements can be efficiently "extracted out". Our second key result shows that, assuming the existence of a leveled FHE with decryption in NC^1, there do not exist extractable semantically secure multilinear encodings–that is, the "Extractable Uber Assumption" is false for every multilinear encoding scheme.

Theorem 3. *[Informally stated] Assume the existence of a leveled FHE with decryption in NC^1. Then no multilinear encodings can satisfy extractable (entropic) semantic security.*

This impossibility result is demonstrated by relying on our construction of indistinguishability obfuscators, showing that if the underlying multilinear encodings satisfy the extractable notion of semantic security, the overall construction will satisfy a "too strong" notion of obfuscation.

Let us mentioned that we do not view Theorem 3 as an indication of the implausibility of "plain" semantical security for multilinear encodings as the notions of extractable and "plain" semantical security are *qualitatively* very different. (Indeed, extractability assumptions have recently been shown to be problematic is various different contexts [HT98, BCCT12, BCPR13, BP13, GGHW13].)

1.5 Outline of the Paper

We provide some preliminaries in Section 2. We define semantical security of multilinear (aka graded) encodings in Section 3. Our construction of an indistinguishability obfuscator is provided in Section 4 and its proof of security is found in Section 5. We finally present the impossibility result for extractable semantical security in Section 6.

2 Preliminaries

Let $\mathbb{N}$ denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \dots, n\}$. Let $\mathbb{Z}$ denote the integers, and $\mathbb{Z}_p$ the integers modulo p. Given a string x, we let $x[i]$, or equivalently x_i, denote the i-th bit of x. For a matrix M, we let $M[i, j]$ denote the entry of M in the ith row and jth column. We use $\mathbf{e}_k$ to denote the vector that is 1 in position k, and 0 in all other positions. The length of $\mathbf{e}_k$ is generally clear from the context. We use $I_{w \times w}$ to denote the identity matrix with dimension $w \times w$.

By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, $M(x)$ represents the distribution of outputs of $M(x)$ when the random tape is chosen uniformly. An oracle algorithm M^O is a machine M that gets oracle access to another machine O, that is, it can access O's functionality as a black-box.

By $x \leftarrow S$, we denote an element x is sampled from a distribution S. If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F. To denote the ordered sequence in which the experiments happen we use semicolon, e.g. $(x \leftarrow S; (y, z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot, \cdot)$ denotes a predicate, then $\Pr[x \leftarrow S; (y, z) \leftarrow A(x) : p(y, z)]$ is the probability that the predicate $p(y, z)$ is true in the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$. The notation $\{(x \leftarrow S; (y, z) \leftarrow A(x) : (y, z))\}$ denotes the resulting probability distribution $\{(y, z)\}$ generated by the ordered sequence of experiments $(x \leftarrow S; (y, z) \leftarrow A(x))$. We define the support of a distribution $\mathsf{supp}(S)$ to be $\{y : \Pr[x \leftarrow S : x = y] > 0\}$.

2.1 Obfuscation

We recall the definition of indistinguishability obfuscation due to [BGI$^+$01].

Definition 1 (Indistinguishability Obfuscator)**.** *A uniform PPT machine $i\mathcal{O}$ is an* indistinguishability obfuscator *for a class of circuits $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if the following conditions are satisfied*

- **Correctness:** *There exists a negligible function ε such that for every $n \in \mathbb{N}$, for all $C \in \mathcal{C}_n$, for all inputs $x \in \{0,1\}^n$, we have*

$$\Pr[C' \leftarrow i\mathcal{O}(C_n) : C'(x) = C_n(x)] = 1 - \varepsilon(n).$$

- **Security:** *For every pair of circuit ensembles $\{C_n^0\}_{n\in\mathbb{N}}$ and $\{C_n^1\}_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$, for every pair of circuits $C_n^0, C_n^1 \in \mathcal{C}_n$ such that $C_n^0(x) = C_n^1(x)$ for all $x \in \{0,1\}^n$ the following holds: For every nuPPT adversary A there exists a negligible function ε such that for all $n \in \mathbb{N}$,*

$$|Pr[C' \leftarrow i\mathcal{O}(C_n^0) : A(C') = 1] - Pr[C' \leftarrow i\mathcal{O}(C_n^1) : A(C') = 1]| \leq \varepsilon(n)$$

2.2 Branching programs for NC^1

We recall the notion of a branching program.

Definition 2 (Matrix Branching Program)**.** *A* branching program *of width w and length m for n-bit inputs is given by a sequence:*

$$BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1})\}_{i=1}^m,$$

where each $B_{i,b}$ is a permutation matrix in $\{0,1\}^{w\times w}$ and $\mathsf{inp}(i) \in [n]$ is the input bit position examined in step i. Then the output of the branching program on input $x \in \{0,1\}^n$ is as follows:

$$BP(x) \overset{def}{=} \begin{cases} 1, & \textit{if } \left(\prod_{i=1}^m B_{i,x[\mathsf{inp}(i)]}\right) \cdot \mathbf{e}_1 = \mathbf{e}_1. \\ 0, & \textit{otherwise} \end{cases}$$

The branching program is said to be oblivious *if $\mathsf{inp} : [m] \to [n]$ is a fixed function, independent of the function being evaluated.*

The above definition differs slightly from the definition of matrix branching programs generally used, which have the slightly stronger requirement that $\prod_{i=1}^n B_{i,x[\mathsf{inp}(i)]} = I_{w\times w}$ when $BP(x)$ is accepting, and $\prod_{i=1}^n B_{i,x[\mathsf{inp}(i)]} = \mathsf{P}_{\mathsf{reject}}$ for some fixed permutation matrix $\mathsf{P}_{\mathsf{reject}} \neq I_{w\times w}$ when $BP(x)$ is rejecting. More generally,

Definition 3. *The branching program is said to have* fixed accept and reject matrices $\mathsf{P}_{\mathsf{accept}}$ *and* $\mathsf{P}_{\mathsf{reject}}$ *if, for all $x \in \{0,1\}^n$,*

$$\prod_{i=1}^m B_{i,x[\mathsf{inp}(i)]} = \begin{cases} \mathsf{P}_{\mathsf{accept}} & \textit{when } BP(x) = 1 \\ \mathsf{P}_{\mathsf{reject}} & \textit{when } BP(x) = 0 \end{cases}$$

We now have the following theorem due to Barrington:

Theorem 4. *([Bar86]) For any depth d and input length n, there exists a length $m = 4^d$, a labeling function $\mathsf{inp} : [m] \to [n]$, an accepting permutation $\mathsf{P}_{\mathsf{accept}}$ with $\mathsf{P}_{\mathsf{accept}} \cdot \mathbf{e}_1 = \mathbf{e}_1$, and a rejecting permutation $\mathsf{P}_{\mathsf{reject}}$ with $\mathsf{P}_{\mathsf{reject}} \cdot \mathbf{e}_1 = \mathbf{e}_k$ where $k \neq 1$ such that, for every fan-in 2 boolean circuit C of depth d and n input bits, there exists an oblivious matrix branching program $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$, of width 5 and length m that computes the same function as the circuit C.*

In particular, every circuit in NC^1 has a polynomial length branching program of width 5. Further, two circuits of the same depth d will have the same fixed accepting and rejecting permutations $\mathsf{P}_{\mathsf{accept}}$ and $\mathsf{P}_{\mathsf{reject}}$, and a fixed labelling function $\mathsf{inp} : [m] \to [n]$.

3 Semantically Secure Graded Encoding Schemes

In this section we define what it means for a graded encoding scheme to be semantically secure. We start by recalling the notion of graded encoding schemes due to Garg, Gentry and Halevi [GGH13a].

3.1 Graded Encoding Schemes

Graded (multilinear) encoding schemes were originally introduced in the work of Garg, Gentry and Halevi [GGH13a]. Just as [BR14, BGK$^+$13], we here rely on "set-based" (or "asymmetric") graded encoding; these were originally called "generalized" graded encodings in [GGH13a]. Following [GGH$^+$13b, BGK$^+$13] and the notion of "multilinear jigsaw puzzles" from [GGH$^+$13b], we additionally enable anyone with the secret parameter to encode *any* elements (as opposed to just *random* elements as in [GGH13a]).

Definition 4 ((k, R)-Graded Encoding Scheme)**.** *A (k, R)-graded encoding scheme for $k \in \mathbb{N}$ and ring R is a collection of sets $\{E_S^\alpha : \alpha \in R, S \subseteq [k]\}$ with the following properties*

- *For every $S \subseteq [k]$ the sets $\{E_S^\alpha : a \in R\}$ are disjoint.*
- *There are associative binary operations $\oplus$ and $\ominus$ such that for every $\alpha_1, \alpha_2 \in R$, $S \subseteq [k]$, $u_1 \in E_S^{\alpha_1}$ and $u_2 \in E_S^{\alpha_2}$ it holds that $u_1 \oplus u_2 \in E_S^{\alpha_1+\alpha_2}$ and $u_1 \ominus u_2 \in E_S^{\alpha_1-\alpha_2}$ where '$+$' and '$-$' are the addition and subtraction operations in R.*
- *There is an associative binary operation $\otimes$ such that for every $\alpha_1, \alpha_2 \in R$, $S_1, S_2 \subseteq [k]$ such that $S_1 \cap S_2 = \emptyset$, $u_1 \in E_{S_1}^{\alpha_1}$ and $u_2 \in E_{S_2}^{\alpha_2}$ it holds that $u_1 \otimes u_2 \in E_{S_1 \cup S_2}^{\alpha_1 \cdot \alpha_2}$ where '$\cdot$' is multiplication in R.*

Definition 5 (Graded Encoded Scheme)**.** *A graded encoding scheme $\mathcal{E}$ is associated with a tuple of PPT algorithms, $(\mathsf{InstGen}_\mathcal{E}, \mathsf{Enc}_\mathcal{E}, \mathsf{Add}_\mathcal{E}, \mathsf{Neg}_\mathcal{E}, \mathsf{Mult}_\mathcal{E}, \mathsf{isZero}_\mathcal{E})$ which behave as follows:*

- *Instance Generation: $\mathsf{InstGen}_\mathcal{E}$ takes as input the security parameter 1^n and multilinearity parameter 1^k, and outputs secret parameters sp and public parameters pp which describe a (k, R)-graded encoding scheme $\{E_S^\alpha : \alpha \in R, S \subseteq [k]\}$. We refer to E_S^α as the set of encodings of the pair (α, S). We restrict to graded encoding schemes where R is $\mathbb{Z}_p$ and p is a prime exponential in n and k.*
- *Encoding: $\mathsf{Enc}_\mathcal{E}$ takes as input the secret parameters sp, an element $\alpha \in R$ and set $S \subseteq [k]$, and outputs a random encoding of the pair (α, S).*
- *Addition: $\mathsf{Add}_\mathcal{E}$ takes as input the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 + \alpha_2, S)$ if $S_1 = S_2 = S$ and outputs $\perp$ otherwise.*
- *Negation: $\mathsf{Neg}_\mathcal{E}$ takes as input the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 - \alpha_2, S)$ if $S_1 = S_2 = S$ and outputs $\perp$ otherwise.*
- *Multiplication: $\mathsf{Mult}_\mathcal{E}$ takes as input the the public parameters pp and encodings $u_1 \in E_{S_1}^{\alpha_1}, u_2 \in E_{S_2}^{\alpha_2}$, and outputs an encoding of the pair $(\alpha_1 \cdot \alpha_2, S_1 \cup S_2)$ if $S_1 \cap S_2 = \emptyset$ and outputs $\perp$ otherwise.*
- *Zero testing: $\mathsf{isZero}_\mathcal{E}$ takes as input the public parameters pp and an encoding $u \in E_S(\alpha)$, and outputs 1 if and only if $\alpha = 0$ and S is the universe set $[k]$.*

Whenever it is clear from the context, to simplify notation we drop the subscript $\mathcal{E}$ when we refer to the above procedures (and simply call them $\mathsf{InstGen}, \mathsf{Enc}, \ldots$).

Note that the above procedures allow algebraic operations on the encodings in a restricted way. Given the public parameters and encodings made under the sets $\vec{S}$, one can only perform algebraic operations that are allowed by the structure of the sets in $\vec{S}$. We call such operations $\vec{S}$-respecting and formalize this notion as follows:

Definition 6 (Set Respecting Arithmetic Circuits). *For any ring R, $k \in \mathbb{N}$ and $\vec{S} \in (2^{[k]})^n$, we say that an arithmetic circuit C (i.e. gates perform only ring operations $\{+, -, \cdot\}$)is $\vec{S}$-*respecting *if it holds that*

- *Eevery input wire of C is tagged with some set in $\vec{S}$.*
- *For every $+$ and $-$ gate in C, if the tags of the two input wires are the same set S then the output wire of the gate is tagged with S. Otherwise the output wire is tagged with $\perp$.*
- *For every $\cdot$ gate in C, if the tags of the two input wires are sets S_1 and S_2 and $S_1 \cap S_2 = \emptyset$ then the output wire of the gate is tagged with $S_1 \cup S_2$. Otherwise the output wire is tagged with $\perp$.*
- *It holds that the output wire is tagged with the universe set $[k]$.*

We say that a circuit C is weakly $\vec{S}$-respecting *if all the above conditions hold except the last, that is, the output wire may be tagged with some set $T \subseteq [k]$, where T is not necessarily equal to $[k]$. We say that C is* non terminal $\vec{S}$-respecting *if T is a strict subset of $[k]$.*

The following lemma is a simple corollary of the efficient procedures described in Definition 5. It states that given the public parameters and some encodings made under the sets $\vec{S}$, one can efficiently zero test the result of any $\vec{S}$ respecting arithmetic circuit on the elements underlying the encodings.

Lemma 5 (Correctness). *Let $\mathcal{E}$ be a graded encoding scheme. There exists a PPT* Eval *such that for any $k, n, m \in \mathbb{N}$, any $(\mathsf{pp}, \mathsf{sp}) \in \mathsf{InstGen}(1^n, 1^k)$ where* pp *describes a (k, R)-graded encoding scheme,* Eval *on input $1^n, \mathsf{pp}$, and a sequence of encodings $\{u_i\}_{i=1}^m$ where $u_i \in E_{S_i}^{\alpha_i}$, α_i is a ring element and $S_i \subseteq [k]$, and any $\vec{S} = \{S_i\}_{i=1}^m$-respecting arithmetic circuit C, outputs 1 if and only if $C(\{\alpha_i\}_{i=1}^m) = 0$.*

3.2 Semantical Security—A generalized GDDH Assumption

We now turn to defining semantical security of graded encoding schemes. Towards explaining our notion of semantical security, let us first consider a "DDH-like" assumption for (asymmetric) multilinear encodings, similar in spirit to the "graded DDH" assumption of Garg et al [GGH13a] for symmetric multilinear encodings: Consider a distribution D sampling n random elements $\vec{z}$, and let $m_0 = \prod_{i \in [n]} z_i$ be the product of the elements in $\vec{z}$, and $m_1 = z'$ be just a random element. A DDH-like assumption—let us refer to it as the "asymmetric graded DDH assumption (aGDDH)"—would require that encodings of m_0 and m_1 under the "target" set $S = [n]$ are indistinguishable, given encodings of $\vec{z}$ under sets $\vec{T}$, if S is not the disjoint union of the sets in $\vec{T}$ (that is, the set-restrictions prohibit "legally" multiplying all the elements of $\vec{z}$). That is, $\{\mathsf{Enc}(\prod_{i \in [n]} z_i, S), \mathsf{Enc}(z_1, T_1), \mathsf{Enc}(z_2, T_2), \ldots \mathsf{Enc}(z_n, T_n)\}$ and $\{\mathsf{Enc}(z', S), \mathsf{Enc}(z_1, T_1), \mathsf{Enc}(z_2, T_2), \ldots \mathsf{Enc}(z_n, T_n)\}$ are indistinguishable.

We here consider a generalized version of such a DDH-like assumption—similar in spirit to the "uber-assumption" of [BBG05]—where we do not only care about products of all the elements in $\vec{z}$ but also more complicated relations among the elements and sets they are encoded under.

As a first attempt, let us consider arbitrary sets $S, \vec{T}$ and elements $m_0, m_1, \vec{z}$ that come from *any* efficient distribution D so that with overwhelming probability over $(m_0, m_1, \vec{z}) \leftarrow D$, *no* "legal" arithmetic ciruit C can tell apart $(m_0, \vec{z})$ and $(m_1, \vec{z})$; that is, with overwhelming probability over $(m_0, m_1, \vec{z}) \leftarrow D$, for every "legal" arithmetic ciruit C, $C(m_0, \vec{z}) = C(m_1, \vec{z})$. This restriction on D, however, is too strong to capture the aGDDH distribution: With high probability over $m_0, m_1, \vec{z}$, there always exists *some* legal arithmetic circuit C such that $C(m_0, \vec{z}) \neq C(m_1, \vec{z})$.[17]

To overcome these issues, we consider a slightly more general class of distributions D, where we switch the order of the quantifiers: Roughly speaking, an efficient distribution D is called "valid" if

[17]Consider a very simple aGDDH instance, where $|\vec{z}| = 2$, $T_1 = T_2 = S = [k]$. For non-zero z_1, z_2, there always exists some a such that the circuit $C(m, z_1, z_2) = \mathsf{isZero}(m - az_1)$ yields different outputs on input $(m_0, \vec{z})$ and $(m_1, \vec{z})$—namely, $a = z_2$.

for every "legal algebraic attacker"—where a legal algebraic attacker A is a, potentially *computationally unbounded*, algorithm that makes polynomially many "legal" arithmetic queries to the elements on which it operates—with overwhelming probability over $m_0, m_1, \vec{z} \leftarrow D$, A produces *exactly the same* output when operating on $(m_0, \vec{z})$ and $(m_1, \vec{z})$. That is, although for most $m_0, m_1, \vec{z}$ there may exists some legal arithmetic circuit that distinguishes them (as is the case for the aGDDH distribution), such a circuit cannot be found using polynomially many queries (but otherwise being computationally unbounded).

It is instructive to note that the aGDDH distribution mentioned above indeed is "valid": any legal arithmetic query is a circuit $C(m, \vec{z})$ of the form $\mathsf{isZero}(a \cdot m + p(\vec{z}))$ where $p(\cdot)$ is a polynomial of degree at most $n-1$. If $a = 0$ and $p(\cdot)$ is the zero-polynomial, then clearly the circuit evaluates to 1. If either $a = 1$ or $p(\cdot)$ is a non-zero polynomial, then no matter whether $m = m_0$ or $m = m_1$, C is evaluating a non-zero polynomial of degree at most n at a random point; by the Schwartz-Zippel lemma, with very high probability (proportional to the field size), both these polynomials will evaluate to a non-zero value, and thus C evaluates to 0. It now follows by a Union Bound that, with overwhelming probability, all the (polynomially many) arithmetic queries made by some algebraic attacker will be answered as above (and thus in an identically the same way), and thus the attacker will produce identically the same output.

It is also worthwhile to note why we restrict the algebraic attacker to only make polynomially many (or subexponentially many) queries; otherwise, it could simply go over all legal arithemetic circuits and the notion would collapse down to the notion considered in the "first attempt" above.

As mentioned in the introduction, for our purposes, we require security to hold in a *constant*-message settings, where m_0,m_1, and S are replaced by *constant-length* vectors $\vec{m_0}, \vec{m_1}, \vec{S}$. We now turn to formalizing the notion of semantically secure graded encodings. We start by formally defining an *algebraic adversary.* Such an adversary, when given a set of elements and sets is restricted to making "legal" arithmetic queries to these elements. We formalize this restriction by considering adversaries that interact with the following oracle.

Definition 7 (Oracle $\mathcal{M}$)**.** *Let $\mathcal{M}$ be an oracle which operates as follows:*

- *$\mathcal{M}$ gets as initial input a ring R, $k \in \mathbb{N}$ and list L of m pairs $\{(\alpha_i, S_i)\}_{i=1}^m$, $\alpha \in R$ and $S \subseteq [k]$.*
- *Every oracle query to $\mathcal{M}$ is an arithmetic circuit $C : R^m \to R$. When queried with C, $\mathcal{M}$ checks whether C is a $\vec{S}$-respecting arithmetic circuit where $\vec{S} = \{S_i\}_{i=1}^m$. If not, $\mathcal{M}$ outputs $\perp$. Otherwise, $\mathcal{M}$ computes C on $\{\alpha_i\}_{i=1}^m$ and outputs 1 if and only if the output of C is zero, and outputs 0 otherwise.*

We turn to formalizing what it means for a distribution over $(\vec{m_0}, \vec{m_1}, \vec{z})$ to be *valid* with respect to the sets $(\vec{S}, \vec{T})$. We define such a distribution through the notion of a $(\vec{S}, \vec{T})$-respecting message sampler.

Definition 8 (Respecting Message Sampler)**.** *Let $\mathcal{E}$ be a graded encoding scheme, $q(\cdot)$, $k(\cdot)$ and $c(\cdot)$ be polynomials and $\{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}}$ be an ensemble where $\vec{S}_n$ is a sequence of $c(n)$ sets and $\vec{T}_n$ is a sequence of $q(n)$ sets $\subseteq [k]$. We say that a nuPPT M is a $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$-*respecting message sampler *(or* valid w.r.t. $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$*) if*

- *M on input 1^n and a public parameter pp computes the ring R associated with pp and next based on only 1^n and R generates and outputs a pair $(\vec{m_0}, \vec{m_1})$ of sequences of $c(n)$ ring elements and a sequence $(\vec{m_0}, \vec{m_1})$ of $q(n)$ ring elements;*
- *For every polynomial $p(\cdot)$, there exists a negligible function ε such that for every security parameter $n \in \mathbb{N}$, every $(\mathsf{sp}, \mathsf{pp}) \in \mathsf{InstGen}(1^n, 1^{k(n)})$, every (computationally unbounded) oracle machine A*

that makes at most $p(n)$ oracle queries[18] *(called the* algebraic adversary*)*

$$Pr[(\vec{m_0}, \vec{m_1}, \vec{z}) \leftarrow M(1^n, \mathsf{pp}) : A^{\mathcal{M}(\mathsf{pp}, \vec{p_0})}(1^n) \neq A^{\mathcal{M}(\mathsf{pp}, \vec{p_1})}(1^n)] \leq \varepsilon(n)$$

where $\vec{p_b} = \{(m_b[i], S_i)\}_{i=1}^{c(n)}, \{(z[i], T_i)\}_{i=1}^{q(n)}$.

Let us comment that Definition 8 allows the message sampler M to select $\vec{m_0}, \vec{m_1}, \vec{z}$ based on the ring $R = \mathbb{Z}_p$; note that this is needed even to model the aGDDH assumption (or else we could not define what it means to pick a uniform element in the ring). (On the other hand, to make the notion of semantic security as weak as possible, we prevent the message selection to depend on pp in any other way, although it also seems reasonable to consider message samplers that do depend on pp).

We can now define what it means for a graded encoding scheme to be semantically secure. Roughly speaking, we require that for any sequence of sets $(\vec{S}, \vec{T})$, and any $(\vec{S}, \vec{T})$-respecting message sampler M, encodings of $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$ under the sets $(\vec{S}, \vec{T})$ are indistinguishable, when $(\vec{m}_0, \vec{m}_1, \vec{z})$ is sampled by M.

Definition 9 (Semantic Security)**.** *Let $\mathcal{E}$ be a graded encoding scheme and $q(\cdot)$, $k(\cdot)$ and $c(\cdot)$ be polynomials. We say a graded encoding scheme $\mathcal{E}$ is* (q, k, c)-semantically secure *if for every ensemble $\{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}}$ where $\vec{S}_n \subseteq [k(n)]^{c(n)}$ and $\vec{T}_n \subseteq [k(n)]^{q(n)}$, every $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$-respecting message sampler M and nuPPT adversary A, there exists a negligible function ϵ such that for every security parameter $n \in \mathbb{N}$,*

$$|Pr[\boldsymbol{Output}_0(q, k, c, A, M, n, (\vec{S}_n, \vec{T}_n)) = 1] - Pr[\boldsymbol{Output}_1(q, k, c, A, M, n, (\vec{S}_n, \vec{T}_n)) = 1]| \leq \epsilon(n)$$

where $\boldsymbol{Output}_b(q, k, c, A, M, n, (\vec{S}_n, \vec{T}_n))$ is A's output in the following game:

- *Let* $(\mathsf{sp}, \mathsf{pp}) \leftarrow \mathsf{InstGen}(1^n, 1^{k(n)})$.
- *Let* $\vec{m_0}, \vec{m_1}, \vec{z} \leftarrow M(1^n, \mathsf{pp})$.
- *Let* $\vec{u_b} \leftarrow \{\mathsf{Enc}(\mathsf{sp}, \vec{m_0}[i], \vec{S}_n[i])\}_{i=1}^{c(n)}, \{\mathsf{Enc}(\mathsf{sp}, \vec{z}[i], \vec{T}_n[i])\}_{i=1}^{q(n)}$.
- *Finally, run* $A(1^n, \mathsf{pp}, \vec{u_b})$.

We say that $\mathcal{E}$ is (constant-message) semantically secure *if it is (q, k, c)-semantically secure for all polynomials $q(\cdot)$, $k(\cdot)$ and all constants $c \in \mathbb{N}$; we say that $\mathcal{E}$* multi-message semantically secure *if the above holds also for all polynomials $c(\cdot)$.*

Semantical Security w.r.t. Restricted Classes of Message Samplers For our specific construction of indistinguishability obfuscators it suffices to assume the existence of *semantically secure encodings w.r.t. restricted classes of message samplers M*, where the $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$-respecting condition on M is replaced by some stronger restriction on M. It particular, it suffices to restrict to message samplers M that induce a *high-entropy* distribution over $\vec{m_0}, \vec{m_1}, \vec{z}$—not only the individual elements have high min-entropy but also any element computed by applying a "non-terminal" sequence of legal arithmetic operations to $\vec{m_b}, \vec{z}$ (for $b \in \{0, 1\}$). More precisely, we say that a M is a *$((q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}}), H)$-entropic respecting message sampler* if M is $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$-respecting and additionally:

- For every security parameter n, every $\mathsf{pp} \in \mathsf{InstGen}(1^n, 1^{k(n)})$ describing a ring R, every non-terminal $(\vec{S}_n, \vec{T}_n)$-respecting arithmetic circuit C that computes a non-zero polynomial in its inputs, it holds that for $b \in \{0, 1\}$,

$$H_\infty(C(\vec{m_b}, \vec{z})) \geq H(\log |R|)$$

where $(\vec{m_0}, \vec{m_1}, \vec{z}) \leftarrow M(1^n, \mathsf{pp})$.

[18]Our proofs work even if the algebraic adversary A makes subexponentially many oracle queries.

We here focus on "very" high entropy message samplers, where $H(n) = n - O(\log n)$, and refer to encoding schemes satisfying semantical security w.r.t. such restricted message samplers as *entropic semantically secure.*

Additionally, for our purposes, we may consider even more restricted types of message samplers. In particular, where:

1. The total number of elements sampled by M is $O(k(n))$[19], and furthermore, the number of encodings under any given set is constant;
2. Each individual element sampled is statistically close to a uniform ring element;
3. Elements sampled are "almost" pair-wise independent: each pair of elements encoded is statistically close to two uniform ring elements;
4. The sets $\vec{S}, \vec{T}$ are pairs of indices $\{i, j\}$, $i, j \in [k]$.

Property 1 is in analogy with previous assumptions over multilinear encodings (e.g., the GDDH assumption of [GGH13a]), where the number of encodings released is tightly bound by the multilinearity level k. Property 2, 3 are again natural abstractions of what happens in the GDDH assumption (property 3 is a relaxation of the independence, as opposed to just pair-wise independence, property satisfied by the GDDH assumption). Property 4 implies that (if we consider an algebraic attacker) exactly $k/2$ multiplications on the elements must be performed before a zero-testing can be done; combined with the above entropic message sampler condition, this implies that any "legal" arithmetic circuit of multiplicative degree smaller than $k/2$ produces a high-entropy element when applied to the sampled elements[20] For lack of a better name, we refer to encoding schemes satifying (entropic) semantical security w.r.t. such restricted message samplers as *(entropic) semantically secure with respect to simple message samplers.*

Finally, for specific *applications* of obfuscation, we may further restrict the class of message sampler for which semantical security must hold. Furthermore, for many such applications, it now becomes easy to check whether a message sampler in the class is valid, and thus semantical security w.r.t. to this class of message samplers is an efficiently falsifiable assumption. More generally, such applications of indistinguishability obfuscation can be based on a "meta-falsfiable" variant of semantical security where the attacker not only needs to come up with a valid message sampler (for which it can distinguish encodings) but also a *mathematical proof* (in some appropriate language) that the message-sampler is valid.

A Slightly Weaker Notion of Semantical Security For our results it suffices to use a (seemingly) stronger notion of a "valid" message sampler (and thus a weaker notion of semantical security): instead of requiring pointwise indistinguishability by algebraic attackers, we require that, with overwhelming probability, the output of arithmetic circuits is a function of only the circuit (and not the input on which it operates); indeed, in our proof we demonstrate exactly this. More precisely, bullet 2 in the Definition 8 is replaced by:

- There exists a negligible function ε such that for every security parameter $n \in \mathbb{N}$, every $(\mathsf{sp}, \mathsf{pp})$ in the support of $\mathsf{InstGen}(1^n, 1^{k(n)})$, there exists a function f such that for every $(\vec{S}, \vec{T})$-respecting arithmetic circuit C,

$$Pr[(\vec{m_0}, \vec{m_1}, \vec{z}) \leftarrow M(1^n, \mathsf{pp}) : C(\vec{m_0}, \vec{z}) = C(\vec{m_1}, \vec{z}) = f(C)] \geq 1 - \varepsilon(n)$$

[19]We thank Hoeteck Wee for encouraging us to make this explicit.

[20]We thank Shai Halevi for this observation (and more generally for suggesting that we consider the output of low-degree arithmetic circuits as an alternative to our entropic condition.).

It follows by a Union Bound (over the number of queries made by an algebraic attacker) that this alternative notion of a "valid" message sampler implies the one from Definition 8. Note that for this argument to go through it is important that the answer to each query (with overwhelming probability) is independent of $\vec{m_0}, \vec{m_1}, \vec{z}$, or else, the answers to earlier queries may help the algebraic attacker to find a circuit that distinguishes $\vec{m_0}, \vec{z}$ and $\vec{m_1}, \vec{z}$.

4 Construction of an Indistinguishability Obfuscator

In this section, we describe our construction of an indistinguishability obfuscator $i\mathcal{O}$. We will prove its security in Section 5, based on the security notions defined above.

As in previous works [GGH+13b, BR14, BGK+13], the strategy for our construction will be to convert an NC^1 circuit into an oblivious matrix branching program, apply Kilian's randomization technique to the matrices, and then encode these matrices using the graded encoding scheme. The encoding will be using a so-called "straddling set system" (as in [BGK+13]) that will enforce that any arithmetic circuit operating on these encodings can be decomposed into a sum of terms such that each term can be expressed using only encodings that come from one branch of the branching program (more specifically, from the path through the branching program corresponding to evaluating a particular input x to the branching program).

The biggest change from previous work is that before randomizing and encoding the branching program, we double its width by chaining a dummy branching program to it that computes the constant 1, and then add a branch at the very start that chooses whether to use the true program or the dummy, based on a "switch".

At a high level, to show indistinguishability of obfuscations of C_1 and C_2, our strategy will be to obfuscate the branching program for C_1 together with the dummy, and then, matrix by matrix, replace the dummy branching program with the branching program for C_2. Once the entire dummy branching program has been replaced by C_2, we flip the "switch" so that the composite branching program now computes the branching program for C_2. We then replace the branching program for C_1 with C_2, matrix by matrix, so that we have two copies of the branching program for C_2. We now flip the "switch" again, and finally restore the dummy branching program, so that we end up with one copy of C_2 and one copy of the dummy.

In this way, we transition from an obfuscation of C_1 to an obfuscation of C_2, while only changing a small piece of the obfuscation in each step, namely a single level of the underlying branching program. We will later show, in the following section, that each step of the transitions must be indistinguishable based on our hardness assumption. In particular, we show that no algebraic adversary can distinguish between two hybrids, and thus the two distributions should be computationally indistinguishable based on our assumption.

4.1 Merging Branching Programs

We now describe a method Merge for combining two branching programs together to create a composite branching program of double width, in a way that enables switching by changing only a small number of matrices.

Construction 1 (Merging branching programs). *Let $BP_0 = \{\mathsf{inp}(i), B^0_{i,0}, B^0_{i,1}\}_{i=1}^m$ and $BP_1 = \{\mathsf{inp}(i), B^1_{i,0}, B^1_{i,1}\}_{i=1}^m$ be oblivious matrix branching programs, each of width w and length m for n input bits. (We assume that the same labelling function $\mathsf{inp} : [m] \to [n]$ is used for each of BP_0 and BP_1.)*
Define branching programs $\widehat{BP}_0 = \{\mathsf{inp}'(i), \hat{B}^0_{i,0}, \hat{B}^0_{i,1}\}_{i=1}^{m+2}$ and $\widehat{BP}_1 = \{\mathsf{inp}'(i), \hat{B}^1_{i,0}, \hat{B}^1_{i,1}\}_{i=1}^{m+2}$, each of

width $2w$ and length $m+2$ on l input bits, where:

$$\mathsf{inp}'(i) \overset{def}{=} \begin{cases} 1, & \textit{when } i = 1 \\ \mathsf{inp}(i-1), & \textit{when } 2 \le i \le m+1 \\ 1, & \textit{when } i = m+2 \end{cases}$$

and, for all levels except the first and the last, $\widehat{BP}_0$ and $\widehat{BP}_1$ agree, given by:

$$\hat{B}^0_{i,b} = \hat{B}^1_{i,b} \overset{def}{=} \begin{pmatrix} B^0_{(i-1),b} & 0 \\ 0 & B^1_{(i-1),b} \end{pmatrix} \quad \textit{for all } 2 \le i \le m+1 \textit{ and } b \in \{0,1\}$$

and the first and last levels are given by:

$$\begin{aligned} \hat{B}^0_{1,b} &= \hat{B}^0_{m+2,b} = I_{2w\times 2w} && \textit{for } b \in \{0,1\} \\ \hat{B}^1_{1,b} &= \hat{B}^1_{m+2,b} = \begin{pmatrix} 0 & I_{w\times w} \\ I_{w\times w} & 0 \end{pmatrix} && \textit{for } b \in \{0,1\} \end{aligned}$$

We define Merge *so that* $\mathsf{Merge}(BP_0, BP_1, 0) = \hat{BP}_0$ *and* $\mathsf{Merge}(BP_0, BP_1, 1) = \hat{BP}_1$.

We will show that $\hat{BP}_0$ and $\hat{BP}_1$ are matrix branching programs that compute the same functions as BP_0 and BP_1 respectively, with the additional feature that $\hat{BP}_0$ and $\hat{BP}_1$ differ from each other in only two levels, namely the first and the last. Further, since inp' does not depend on the function being computed, $\hat{BP}_0$ and $\hat{BP}_1$ are *oblivious* matrix branching programs.

Accordingly, with respect to $\mathsf{Merge}(BP_0, BP_1, b)$ we will often use the phrase *active branching program* to refer to BP_b.

Claim 6. *For $BP_0 = \{\mathsf{inp}(i),\ B^0_{i,0}, B^0_{i,1}\}_{i=1}^m$ and $BP_1 = \{\mathsf{inp}(i), B^1_{i,0}, B^1_{i,1}\}_{i=1}^m$ each of width w and length m on n input bits, define $\widehat{BP}_0$ and $\widehat{BP}_1$ as above. Then, for each $b \in \{0,1\}$, $x \in \{0,1\}^n$,*

$$\prod_{i=1}^{m+2} \widehat{B}^b_{i,x[\mathsf{inp}'(i)]} = \begin{pmatrix} \prod_{i=1}^m B^b_{i,x[\mathsf{inp}(i)]} & 0 \\ 10 & \prod_{i=1}^m B^{1-b}_{i,x[\mathsf{inp}(i)]} \end{pmatrix}$$

Proof. We observe that $\widehat{BP}_0$ and $\widehat{BP}_1$ agree on each level except the first and last, that is,

$$\widehat{B}^0_{i,b} = \widehat{B}^1_{i,b} = \begin{pmatrix} B^0_{(i-1),b} & 0 \\ 0 & B^1_{(i-1),b} \end{pmatrix} \qquad \forall \quad i : 2 \le i \le m+1, \quad b \in \{0,1\}$$

Then we have, for any $x \in \{0,1\}^n$,

$$\begin{aligned} \prod_{i=2}^{m+1} \widehat{B}^0_{i,x[\mathsf{inp}'(i)]} = \prod_{i=2}^{m+1} \widehat{B}^1_{i,x[\mathsf{inp}'(i)]} &= \prod_{i=2}^{m+1} \begin{pmatrix} B^0_{(i-1),x[\mathsf{inp}'(i)]} & 0 \\ 0 & B^1_{(i-1),x[\mathsf{inp}'(i)]} \end{pmatrix} \\ &= \prod_{i=1}^{m} \begin{pmatrix} B^0_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & B^1_{i,x[\mathsf{inp}(i)]} \end{pmatrix} \\ &= \begin{pmatrix} \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} \end{pmatrix} \end{aligned}$$

Where the change of indices in the second step follows because $\mathsf{inp}'(i) = \mathsf{inp}(i-1)$ when $2 \le i \le m+1$. We now consider the two case for $b \in \{0,1\}$.

Case 1: (b = 0)
In this case,

$$\begin{aligned}\prod_{i=1}^{m+2} \widehat{B}^0_{i,x[\mathsf{inp}'(i)]} &= I_{2w\times 2w} \cdot \begin{pmatrix} \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} \end{pmatrix} \cdot I_{2w\times 2w} \\ &= \begin{pmatrix} \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} \end{pmatrix}\end{aligned}$$

as required.
Case 2: (b = 1)
In this case,

$$\begin{aligned}\prod_{i=1}^{m+2} \widehat{B}^1_{i,x[\mathsf{inp}'(i)]} &= \begin{pmatrix} 0 & I_{w\times w} \\ I_{w\times w} & 0 \end{pmatrix} \cdot \begin{pmatrix} \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} \end{pmatrix} \cdot \begin{pmatrix} 0 & I_{w\times w} \\ I_{w\times w} & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} \\ \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & I_{w\times w} \\ I_{w\times w} & 0 \end{pmatrix} \\ &= \begin{pmatrix} \prod_{i=1}^m B^1_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^0_{i,x[\mathsf{inp}(i)]} \end{pmatrix}\end{aligned}$$

as required. □

Claim 7. *For all BP_0 and BP_1 each of width w and length m on n input bits, for each $b \in \{0,1\}$, for all $x \in \{0,1\}^n$,*

$$\mathsf{Merge}(BP_0, BP_1, b)(x) = BP_b(x)$$

Proof. Let $BP_0 = \{\mathsf{inp}(i), B^0_{i,0}, B^0_{i,1}\}_{i=1}^m$ and $BP_1 = \{\mathsf{inp}(i), B^1_{i,0}, B^1_{i,1}\}_{i=1}^m$. Define $\hat{BP}_0 = \mathsf{Merge}(BP_0, BP_1, 0)$ and $\hat{BP}_1 = \mathsf{Merge}(BP_0, BP_1, 1)$ as above. We observe that for any $x \in \{0,1\}^n$,

$$\begin{aligned}&\mathsf{Merge}(BP_0, BP_1, b)(x) = 1 \\ \iff& \Big(\prod_{i=1}^{m+2} \widehat{B}^b_{i,x[\mathsf{inp}'(i)]}\Big) \cdot \mathbf{e}_1 = \mathbf{e}_1 \\ \iff& \begin{pmatrix} \prod_{i=1}^m B^b_{i,x[\mathsf{inp}(i)]} & 0 \\ 0 & \prod_{i=1}^m B^{1-b}_{i,x[\mathsf{inp}(i)]} \end{pmatrix} \cdot \mathbf{e}_1 = \mathbf{e}_1 \qquad \text{(from Claim 6)} \\ \iff& \Big(\prod_{i=1}^{m} B^b_{i,x[\mathsf{inp}(i)]}\Big) \cdot \mathbf{e}_1 = \mathbf{e}_1 \\ \iff& BP_b(x) = 1\end{aligned}$$

Thus $\mathsf{Merge}(BP_0, BP_1, b)(x) = BP_b(x)$. □

The following claim illustrates some useful properties of the Merge procedure that we will use later. Firstly it notes that changing the bit Merge gets as input changes only the "switch" matrices in the first and last level of the program Merge outputs. Secondly, changing one level of a program Merge gets as input changes the output program in one level only. Finally, the first column of the output matrix of the widened program output by Merge depends only on the first column of the output matrix of the active program. The claim follows by observing the definition of Merge.

Claim 8. *Let BP_0 and BP_1 be length m, width w branching programs, with input length n.*

- $\mathsf{Merge}(BP_0, BP_1, 0)$ *and* $\mathsf{Merge}(BP_0, BP_1, 1)$ *differ in only 4 matrices, the matrices corresponding to the first and last level.*

- *Let* BP_1' *be a length* m *branching program that differs from* BP_1 *in only the* i^{th} *level for some* $i \in [m]$. *Then for both* $b \in \{0,1\}$, $\mathsf{Merge}(BP_0, BP_1, b)$ *and* $\mathsf{Merge}(BP_0, BP_1', b)$ *also differ only in the* i^{th} *level. A similar statement holds for branching programs* BP_0' *that differ from* BP_0 *in only one level.*

- *For any* $b \in \{0,1\}$, *let* $BP = \mathsf{Merge}(BP_0, BP_1, b)$, *and* ${\mathsf{P_{out}}}^{BP}(\cdot)$ *and* ${\mathsf{P_{out}}}^{BP_b}(\cdot)$ *be the functions computing the output matrices on a given input for* BP *and* BP_b *respectively. Then for every input* $x \in \{0,1\}^n$,
$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{extend}(\mathsf{col}_1({\mathsf{P_{out}}}^{BP_b}(x)))$$
where extend *extends a length* w *vector by appending* w *zeroes to the end.*

4.2 Randomizing Branching Programs

We now describe Kilian's randomization technique [Kil88] for a branching program, adapted to our setting, by defining a process Rand that randomizes the matrices of a branching program BP. We will decompose the randomization into two parts, Rand^B and Rand^α, defined below, and define Rand as their composition.

Definition 10 (Rand^B)**.** *Let* $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ *be a branching program of width* w *and length* m, *with length-*n *inputs. Let* p *be a prime exponential in* n. *Then the process* $\mathsf{Rand}^B(BP, p)$ *samples* m *random invertible matrices* $R_1, R_2, \dots, R_m \in Z_p^{w\times w}$ *uniformly and independently, and computes*
$$\tilde{B}_{i,b} = R_{(i-1)} \cdot B_{i,b} \cdot R_i^{-1} \quad \text{for every } i \in [m], \text{ and } b \in \{0,1\}$$
where R_0 *is defined as* $I_{w\times w}$, *and*
$$\mathbf{t} = R_m \cdot \mathbf{e}_1$$
Rand^B *then outputs*
$$(\{\tilde{B}_{i,b}\}_{i\in[m], b\in\{0,1\}}, \mathbf{t}, p)$$

Definition 11 (Rand^α)**.** *Let* $(\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}, p)$ *be the output of* $\mathsf{Rand}^B(BP, p)$ *as defined above. On this input,* $\mathsf{Rand}^\alpha(\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, p)$ *samples* $2m$ *non-zero scalars* $\{\alpha_{i,b} \in \mathbb{Z}_p : i \in [m], b \in \{0,1\}\}$ *uniformly and independently, and outputs*
$$(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$$

Definition 12 (Rand)**.** *Let* $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ *be a branching program of width* w *and length* m, *with length-*n *inputs. Let* p *be a prime exponential in* n. *Then we define* $\mathsf{Rand}(BP, p)$ *to be:*
$$\begin{aligned}\mathsf{Rand}(BP, p) &= (\mathsf{Rand}^\alpha(\mathsf{Rand}^B(BP, p))) \\ &= (\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})\end{aligned}$$
Where $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$ *are as computed in the executions of* Rand^α *and* Rand^B.

Execution of a randomized branching program: To compute $BP(x)$ from the output of $\mathsf{Rand}(BP, p)$, given some input labelling function $\mathsf{inp} : [m] \to [n]$, and $x \in \{0,1\}^n$, we compute
$$\mathsf{Out}(x) = (\prod_{i=1}^m \alpha_{i,x[\mathsf{inp}(i)]} \cdot \tilde{B}_{i,x[\mathsf{inp}(i)]}) \cdot \mathbf{t}$$

Where $\mathsf{Out} \in Z_P^w$ is a $w \times 1$ vector. The intermediate multiplications cause each R_i^{-1} to cancel each R_i, and $R_0 = I_{w \times w}$, so the above computation can also be expressed as:

$$\mathsf{Out}(x) = (\prod_{i=1}^{m} \alpha_{i,x[\mathsf{inp}(i)]} \cdot B_{i,x[\mathsf{inp}(i)]}) \cdot \mathbf{e}_1$$

When $BP(x) = 1$, we have that

$$\prod_{i=1}^{m} \alpha_{i,x[\mathsf{inp}(i)]} \cdot B_{i,x[\mathsf{inp}(i)]} \cdot \mathbf{e}_1 = (\prod_{i=1}^{m} \alpha_{i,x[\mathsf{inp}(i)]}) \cdot \mathbf{e}_1$$

When $BP(x) = 0$, we have that

$$\prod_{i=1}^{m} \alpha_{i,x[\mathsf{inp}(i)]} \cdot B_{i,x[\mathsf{inp}(i)]} \cdot \mathbf{e}_1 = (\prod_{i=1}^{m} \alpha_{i,x[\mathsf{inp}(i)]}) \cdot \mathbf{e}_k$$

for $k \neq 1$. Hence, to compute $BP(x)$, we compute $\mathsf{Out}(x)$ and output 0 if $\mathsf{Out}(x)[1] = 0$, and 1 otherwise.

Simulating a randomized branching program: Previous works ([BGK$^+$13, BR14]) followed [Kil88] to show how to simulate the distribution of any single path corresponding to an input x using just $BP(x)$. However, the simulator required that branching programs have unique accept and reject matrices $\mathsf{P}_{\mathsf{accept}}$ and $\mathsf{P}_{\mathsf{reject}}$.

We would also like a theorem, along the lines of [Kil88], that shows that any single path through a randomized branching program BP corresponding to an input x can be simulated knowing just the accept/reject behavior of BP on x (i.e. by knowing whether $BP(x) = 1$).

In our setting, however, branching programs only meet the relaxed requirement that the output matrix $\mathsf{P}_{\mathsf{out}}(x)$ computed by evaluating BP on input x satisfies $\mathsf{P}_{\mathsf{out}}(x) \cdot \mathbf{e}_1 = \mathbf{e}_1 \iff BP(x) = 1$. There can thus be multiple accept and reject matrices, and the particular accept or reject matrix output by BP can depend both on x and on the specific implementation of BP (and not simply its accept/reject behavior). In contrast, in previous works, because $\mathsf{P}_{\mathsf{accept}}$ and $\mathsf{P}_{\mathsf{reject}}$ were unique, knowing just the accept/reject behavior of BP on x also determines $\mathsf{P}_{\mathsf{out}}(x)$.

What we will show is that, for the particular randomization scheme chosen above, we can simulate any single path through a randomized branching program BP corresponding to an input x without knowing the exact accept/reject matrix $\mathsf{P}_{\mathsf{out}}(x)$, but rather just knowing the first column $\mathsf{p}_{\mathsf{out}}(x) = \mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x))$.

This will be sufficient for our applications, because the class of branching programs we randomize will have the property that there are fixed columns $\mathsf{p}_{\mathsf{accept}}$ and $\mathsf{p}_{\mathsf{reject}} \in \mathbb{Z}_p^w$ such that for all $x \in \{0,1\}^n$, if $BP(x) = 1$ then $\mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x)) = \mathsf{p}_{\mathsf{accept}}$, and if $BP(x) = 0$ then $\mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x)) = \mathsf{p}_{\mathsf{reject}}$. In the case of such programs, $\mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x))$ is determined solely by $BP(x)$, and not the particular implementation of BP. Thus, for these programs, we can simulate given only $BP(x)$.

Before we show this theorem, we define notation for a path through a branching program corresponding to an input x.

Definition 13 (proj_x). *Let* $\mathsf{inp} : [m] \to [n]$ *be an input labelling function, and, for any* $x \in \{0,1\}^n$, *define* proj_x*, relative to* inp*, such that for any branching program* BP *with labelling function* inp*, for any prime* $p \in \mathbb{N}$*, and for any* $(\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}^B(BP, p)$

$$\mathsf{proj}_x(\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = (\{\tilde{B}_{i,x[\mathsf{inp}(i)]}\}_{i\in[m]}, \mathbf{t}),$$

that is, proj_x *selects the elements from* $(\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$ *used when evaluating input* x.

We now show a version of Kilian's theorem, adapted to our construction.

Theorem 9. *There exists an efficient simulator* KSim *such that the following holds. Let* $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i\in[m]}$ *be a width-*w *branching program of length* m *on* n *bit inputs, and* p *a prime exponential in* n*. Let* $x \in \{0,1\}^n$ *be an input to BP, and let* $b_i = x[\mathsf{inp}(i)]$ *for each* $i \in [m]$*. Let* $\mathsf{P_{out}}(x) = \prod_{i=1}^m B_{i,b_i}$ *denote the matrix obtained by evaluating* BP *on* x*, and let* $\mathsf{p_{out}}(x) = \mathsf{col}_1(\mathsf{P_{out}}(x))$ *denote the first column of this output. Let* $\mathsf{proj}_x(\mathsf{Rand}^B(BP,p))$ *be defined respecting the labelling function* inp*. Then* $\mathsf{KSim}(1^m, p, \mathsf{p_{out}}(x))$ *is identically distributed to* $\mathsf{proj}_x(\mathsf{Rand}^B(BP,p))$*.*

Proof. We begin by defining $\mathsf{KSim}(1^n, p, BP(x))$ as follows:

- For each i, KSim selects $\tilde{B}_{i,b_i}$ to be a uniformly random invertible matrix in $Z_p^{w\times w}$.
- KSim selects $\mathbf{t} \in \mathbb{Z}_p^w$ solving

$$(\prod_{i\in[m]} \tilde{B}_{i,b_i}) \cdot \mathbf{t} = \mathsf{p_{out}}(x) \tag{1}$$

  where $b_i = x[\mathsf{inp}(i)]$ for each i.
- KSim outputs $\{\{\tilde{B}_{i,b_i}\}_{i\in[m]}, \mathbf{t}\}$

We want to show that the distribution output by KSim matches the real distribution of $\{\{\tilde{B}_{i,b_i}\}_{i\in[m]}, \mathbf{t}\}$ in the output of $\mathsf{Rand}^B(BP,p)$. But from [Kil88], we have the following:

Claim 10. *The distribution of* $\{\{\tilde{B}_{i,b_i}\}_{i\in[m]}, R_m\}$ *can be exactly sampled given* $\mathsf{P_{out}}(x)$*, by sampling* $\{\tilde{B}_{i,b_i}\}_{i\in[m]}$*,* R_m *to be uniformly random and independent invertible matrices in* $\mathbb{Z}_p^{w\times w}$ *subject to*

$$(\prod_{i\in[m]} \tilde{B}_{i,b_i}) \cdot R_m = \mathsf{P_{out}}(x) \tag{2}$$

The above claim implies the following:

Claim 11. *The distribution of* $\{\{\tilde{B}_{i,b_i}\}_{i\in[m]}, R_m\}$ *can be sampled by independently choosing each* $\tilde{B}_{i,b_i}$ *uniform and invertible, and fixing* R_m *solving equation (2).*

Proof. This follows because for every choice of invertible $\tilde{B}_{i,b_i}$, there exists R_m solving equation (2) given by

$$R_m = (\prod_{i\in[m]} \tilde{B}_{i,b_i}))^{-1} \cdot \mathsf{P_{out}}(x) \tag{3}$$

Further, every solution to equation (2) can be represented as invertible $\tilde{B}_{i,b_i}$, and an R_m solving equation (3). Thus choosing a random solution to equation (2) corresponds to independently choosing each $\tilde{B}_{i,b_i}$ uniformly and invertible, and fixing R_m solving equation (3). □

From the above argument, we have that the distribution of $\mathsf{proj}_x(\mathsf{Rand}(BP,p))$ is exactly the same as the distribution produced by independently choosing each $\tilde{B}_{i,b_i}$ uniform and invertible, fixing R_m solving equation (3), setting $\mathbf{t}$ to be the first column of R_m, and outputting $\{\{\tilde{B}_{i,b_i}\}_{i\in[m]}, \mathbf{t}\}$. But note that each column $\mathsf{col}_i(R_m), i \in [w]$ is the unique solution to

$$(\prod_{i\in[m]} \tilde{B}_{i,b_i}) \cdot \mathsf{col}_i(R_m) = \mathsf{col}_i(\mathsf{P_{out}}(x))$$

Thus we have that each $\tilde{B}_{i,b_i}$ is independent, uniform, and invertible, and, using $i = 1$, $\mathbf{t}$ is the unique solution to

$$(\prod_{i\in[m]} \tilde{B}_{i,b_i}) \cdot \mathbf{t} = \mathsf{p}_{\mathsf{out}}(x)$$

and, in particular, that $\mathbf{t}$ is determined by *only* the first column of $\mathsf{P}_{\mathsf{out}}(x)$. Thus, we see that the distribution of $\mathsf{proj}_x(\mathsf{Rand}^B(BP, p))$ is exactly the same as that output by KSim. $\square$

4.3 Choosing a Set System

In this section we will describe how to choose a collection of sets under which to encode a randomized branching program using the graded encoding scheme. Our selection of sets will closely follow [BGK$^+$13], in that we use straddling set systems. However, one difference is that while they use dual input branching programs, we restrict our attention to single-input schemes. As a consequence, the sets will be simpler and consist of fewer elements.

We first define straddling set systems.

Definition 14 (Straddling Set Systems [BGK$^+$13])**.** *A* straddling set system *with n entries is a collection of sets $\mathbb{S}_n = \{S_{i,b} : i \in [n], b \in \{0,1\}\}$ over a universe U, such that:*

$$\bigcup_{i\in[n]} S_{i,0} = \bigcup_{i\in[n]} S_{i,1} = U$$

and for every distinct non-empty sets $C, D \subseteq \mathbb{S}_n$, we have that if:

1. *(Disjoint Sets:) C contains only disjoint sets. D contains only disjoint sets.*
2. *(Collision:) $\bigcup_{S\in C} S = \bigcup_{S\in D} S$*

Then it must be that $\exists b \in \{0,1\}$ such that:

$$C = \{S_{j,b}\}_{j\in[n]} \quad , \quad D = \{S_{j,(1-b)}\}_{j\in[n]}$$

Informally, the guarantee provided by a straddling set system is that only way to exactly cover U using elements from $\mathbb{S}_n$ is to use either all sets $\{S_{i,0}\}_{i\in n}$ or all sets $\{S_{i,1}\}_{i\in n}$. We use a slight variant of their construction, choosing U to be $[2n]$, each $S_{i,0}$ to be one of $\{1,2\},\{3,4\},\ldots,\{2n-1,2n\}$, and each $S_{i,1}$ to be one of $\{1,2n\},\{2,3\},\{4,5\}\ldots,\{2n-2,2n-1\}$.[21] By a proof exactly following [BGK$^+$13], we have that this construction is a straddling set system.

Theorem 12 (Following Construction 1 in [BGK$^+$13])**.** *For every $n \in N$, there exists a straddling set system $\mathbb{S}_n$ with n entries, over a universe U of $2n$ elements; furthermore, each set in the straddling set system has size exactly two.*

We now define the process $\mathsf{SetSystem}$ which takes as input the length m of a branching program, the number of input bits n, and the input labelling function $\mathsf{inp} : [m] \to [n]$ for a branching program. $\mathsf{SetSystem}$ will output the collection of straddling set systems that we will use to encode any branching program of length m on n input bits, with labelling function inp.

[21] In the construction of [BGK$^+$13], $U = [2n-1]$, and each $S_{i,0}$ is one of $\{1\},\{2,3\},\ldots,\{2n-2,2n-1\}$, and each $S_{i,1}$ is one of $\{1,2\},\{3,4\},\ldots,\{2n-1\}$. We could have also worked with this construction, but modify it slightly to ensure that all encodings are under sets of size exactly two.

Execution of $\mathsf{SetSystem}(m, n, \mathsf{inp})$:
We let n_j denote the number of levels that inspect the jth input bit in inp. That is,

$$n_j = |\{i \in [m] : \mathsf{inp}(i) = j\}|$$

For every $j \in [n]$, $\mathsf{SetSystem}$ chooses $\mathbb{S}^j$ to be a straddling set system with n_j entries over a set U_j, such that the sets $U_1, \ldots, U_n$ are disjoint. Let $U = \bigcup_{j\in[n]} U_j$. $\mathsf{SetSystem}$ then chooses S_t be a set of two elements[22], disjoint from U. We associate the set system $\mathbb{S}^j$ with the j'th input bit of the branching program corresponding to inp. $\mathsf{SetSystem}$ then re-indexes the elements of $\mathbb{S}^j$ to match the steps of the branching program as described by inp, so that:

$$\mathbb{S}^j = \{S_{i,b} : \mathsf{inp}(i) = j, b \in \{0, 1\}\}$$

By this indexing, we also have that $S_{i,b} \in \mathbb{S}^{\mathsf{inp}(i)}$ for every $i \in [m]$, for every $b \in \{0, 1\}$.

Let $k = |U \cup S_t|$, and WLOG, assume that the U^js and S_t are disjoint subsets of $[k]$ (otherwise $\mathsf{SetSystem}$ relabels the elements to satisfy this property).

$\mathsf{SetSystem}$ then outputs

$$k, \quad \{S_{i,b}\}_{i\in[m],b\in\{0,1\}}, \quad S_t$$

4.4 Obfuscating Branching Programs

In this section, we will describe a process Obf that obfuscates a given branching program BP. This process will use Rand and $\mathsf{SetSystem}$ as subroutines. The output of Obf will be a randomized width-10 oblivious matrix branching program, encoded under the graded encoding scheme.

Description of $\mathsf{Obf}(BP)$:

Input. Obf takes as input an oblivious permutation branching program $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^{m}$ of width w and length m on n input bits.

Choosing sets. Obf runs $\mathsf{SetSystem}(m, n, \mathsf{inp})$ and receives $k, \{S_{i,b}\}_{i\in[m+2],b\in\{0,1\}}, S_t$.

Initializing the GES. Obf runs $\mathsf{InstGen}(1^n, 1^k)$ and receives secret parameters sp and public parameters pp which describe a (k, R)-graded encoding scheme. We assume the ring R is equal to $\mathbb{Z}_p$ for some p exponential in n and k.

Randomizing BP. Obf executes $\mathsf{Rand}(BP, p)$, and obtains its output, $\{\{\mathsf{inp}(i), \alpha_{i,0}{\cdot}\tilde{B}_{i,0}, \alpha_{i,1}{\cdot}\tilde{B}_{i,1}\}_{i\in[m]}, \mathbf{t}\}$

Output. Obf outputs:

$$\mathsf{pp}, \quad \{\mathsf{inp}(i), \quad \mathsf{Enc}(\mathsf{sp}, \alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,0}), \quad \mathsf{Enc}(\mathsf{sp}, \alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,1})\}_{i\in[m]}, \quad \mathsf{Enc}(\mathsf{sp}, \mathbf{t}, S_t)$$

We also define a generic version of Obf, which we refer to as GObf. Its output will be used to initialize an oracle $\mathcal{M}$ for the idealized version of the graded encoded scheme. $\mathsf{GObf}(BP, \mathsf{pp})$ acts exactly as $\mathsf{Obf}(BP)$, except that it works with a fixed public parameter pp supplied as input, and in the **Output** step, GObf outputs

$$\mathsf{pp}, \quad \{\mathsf{inp}(i), (\alpha_{i,0} \cdot \tilde{B}_{i,0}, S_{i,0}), (\alpha_{i,1} \cdot \tilde{B}_{i,1}, S_{i,1})\}_{i\in[m]}, \quad (\mathbf{t}, S_t)$$

that is, the output before it is encoded under the multilinear encoding scheme.

[22]We make this choice to ensure that every set in the output of $\mathsf{SetSystem}$consists of exactly two indices $\{i, j\}$ for $i, j \in [k]$

4.5 Putting it all together: Obfuscating NC^1 circuits

We now define our indistinguishability obfuscator $i\mathcal{O}$ for NC^1, as follows:

Description of $i\mathcal{O}(C)$:

1. $i\mathcal{O}$ takes as input $C \in \mathsf{NC}^1$, a fan-in 2 circuit with depth d on n input bits. $i\mathcal{O}$ uses Barrington's Theorem to convert C into an oblivious width 5 permutation branching program $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i=1}^m$ of length $m = 4^d$ on n input bits.
2. $i\mathcal{O}$ generates a dummy width-5 branching program $I = \{\mathsf{inp}(i), I_{5\times5}, I_{5\times5}\}_{i=1}^m$ of length m, where each permutation matrix at each level is the identity matrix. $i\mathcal{O}$ then computes $\widehat{BP} = \mathsf{Merge}(BP, I, 0)$.
3. $i\mathcal{O}$ outputs $\mathsf{Obf}(\widehat{BP})$, which yields the public parameter pp for the graded encoding scheme, together with the encoded branching program $\{\mathsf{inp}(i), \mathsf{Enc}(\alpha_{i,0}\cdot\tilde{B}_{i,0}, S_{i,0}), \mathsf{Enc}(\alpha_{i,1}\cdot\tilde{B}_{i,1}, S_{i,1})\}_{i\in[m+2]}, \mathsf{Enc}(\mathbf{t}, S_t)$.

Correctness of $i\mathcal{O}$: In order to compute the output of $C(x)$ given its obfuscation $i\mathcal{O}(C)$, we perform matrix multiplication on the encoded matrices using the functions Add and Mult of the graded encoding scheme. That is, letting $b_i = x[\mathsf{inp}(i)]$ for each $i \in [m+2]$, using the Eval function guaranteed by Lemma 5, we compute the encoding of

$$\mathsf{Out}(x) = (\prod_{i=1}^{m+2} \alpha_{i,b_i} \cdot \tilde{B}_{i,b_i}) \cdot \mathbf{t}$$

and perform isZero on the encoding of $\mathsf{Out}(x)[1]$ (Note we can only apply Eval and Lemma 5 if the above computation is $\vec{S}$-respecting, but we will show that it is momentarily). From the correctness of the underlying randomized branching program, we have that $C(x) = 0 \iff \mathsf{Out}(x)[1] = 0$. Thus, $i\mathcal{O}$ is correct as long as the above computation is a $\vec{S}$-respecting circuit.

Note that when multiplying two matrices M_1 and M_2 encoded under S_1 and S_2 respectively, the multiplication is $\vec{S}$-respecting as long as $S_1 \cap S_2 = \emptyset$. Thus it suffices to show that the sets encoding the matrices being multiplied, namely:

$$S_{1,b_1}, \quad S_{2,b_2}, \quad \ldots, \quad S_{m+2,b_{m+2}}, \quad S_t$$

are all disjoint, and that their union is $[k]$.

Disjointness follows by observing that each of $U_1, U_2, \ldots, U_n, B_t$ is disjoint, and further that for each $j \in [n]$, for any i, i' such that $\mathsf{inp}(i) = \mathsf{inp}(i') = j$, we have that $b_i = b_{i'} = x[\mathsf{inp}(i)]$ and S_{i,b_i} and $S_{i',b_{i'}}$ are both elements of the straddling set system $\mathbb{S}^{\mathsf{inp}(i)}$, so $S_{i,b_i} \cap S_{i',b_{i'}} = \emptyset$.

To show that the union of the sets is $[k]$, we note that

$$(\bigcup_{i=1}^{m+2} S_{i,b_i}) \cup S_t = (\bigcup_{j=1}^{n} \bigcup_{i:\mathsf{inp}(i)=j} S_{i,x[j]}) \cup S_t = (\bigcup_{j=1}^{n} U_j) \cup S_t = [k]$$

by construction. Thus we have that $i\mathcal{O}$ is correct.

5 Proof of Indistinguishability Obfuscation

Theorem 13. *Assume the existence of an multilinear encoding schemes that is entropic semantically secure (with respect to simple message samplers). Then there exists indistinguishability obfuscators the the class of NC^1 circuits. (More specifically, to obfuscate a constant-width branching program of length $m(n)$, we rely on the the existence of $(O(m), O(m), O(1))$-entropic semantically secure multilinear encoding scheme.)*

Proof. We show that the obfuscator defined in Section 4 is an indistinguishability obfuscator for NC^1 circuits. Consider two NC^1 circuit ensembles $\{C_n^0\}_{n\in\mathbb{N}}$ and $\{C_n^1\}_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $C_n^0(x) = C_n^1(x)$. Assume for contradiction there exists a nuPPT distinguisher D and polynomial p such that for infinitely many n, D distinguishes $i\mathcal{O}(C_n^0)$ and $i\mathcal{O}(C_n^1)$ with advantage $1/p(n)$. For any $n \in \mathbb{N}$ let BP_0 and BP_1 be the branching programs of length $m = poly(n)$ obtained by applying Theorem 4 to the circuits C_n^0 and C_n^1 respectively.

We organize the proof in three parts. In the first part we show that if D distinguishes between obfuscations of C_n^0 and C_n^1 then there exists widened branching programs BP and BP' that differ in only few matrices and evaluate the same function such that D distinguishes between $\mathsf{Obf}(BP)$ and $\mathsf{Obf}(BP')$. Furthermore, the first column of the output matrix is the same for BP and BP', and depends only on the output of the program $BP(x) = BP'(x)$. More concretely, there exist vectors v_0 and v_1 such that for all inputs x the first column of the output matrix for both BP and BP' is always $v_{BP(x)}$.

In the second part, we apply the semantic security of the graded encoding scheme used to argue that if D distinguishes $\mathsf{Obf}(BP)$ and $\mathsf{Obf}(BP')$ then there exists an algebraic adversary that does the same. In particular, this adversary can distinguish between the oracles $\mathcal{M}(\mathsf{GObf}(BP, \mathsf{pp}))$ and $\mathcal{M}(\mathsf{GObf}(BP', \mathsf{pp}))$ for some public parameters pp. Finally, in the third part we show that these oracles can be simulated given oracle access to BP (resp. BP') and input v_0 and v_1. This, together with the fact that BP and BP' agree on all inputs, will imply a contradiction and hence prove the theorem.

5.1 Setting up BP and BP' via a Hybrid Argument

Let Hyb_i be a procedure that takes an input two length m branching programs P_0 and P_1 (with the same labeling function) and outputs a "hybrid" length m branching program whose first i levels are identical to the first i levels of P_0 and all the other levels are identical to those of P_1. Formally, let $P_0 = \{\mathsf{inp}(j), B_{j,0}, B_{j,1}\}_{j\in[m]}$ and $P_1 = \{\mathsf{inp}(j), B'_{j,0}, B'_{j,1}\}_{j\in[m]}$.

$$\mathsf{Hyb}_i(P_0, P_1) = \{\mathsf{inp}(j), B_{j,0}, B_{j,1}\}_{j=1}^{i}, \{\mathsf{inp}(j), B'_{j,0}, B'_{j,1}\}_{j=i+1}^{m}$$

For every $n \in \mathbb{N}$ we define hybrid distributions in the following way.

- We start with H_0 which is the obfuscation of the circuit C_n^0.

$$H_0 = i\mathcal{O}(C_n^0) = \mathsf{Obf}(\mathsf{Merge}(BP_0, I, 0))$$

- For $i = 1, 2 \dots m$, let

$$H_i = \mathsf{Obf}(\mathsf{Merge}(BP_0, \mathsf{Hyb}_i(BP_1, I), 0))$$

  We change, one level at a time, the second branching program Merge takes as input from I to BP_1.

- We have that $H_m = \mathsf{Obf}(\mathsf{Merge}(BP_0, BP_1, 0))$. We change the "switch" input to Merge so that the second branching program BP_1 is active.

$$H_{m+1} = \mathsf{Obf}(\mathsf{Merge}(BP_0, BP_1, 1))$$

- For $i = 1, 2 \dots m$, let

$$H_{m+i+1} = \mathsf{Obf}(\mathsf{Merge}(\mathsf{Hyb}_i(BP_1, BP_0), BP_1, 1))$$

  We change the first program Merge takes as input from BP_0 to BP_1, one level at a time as before.

- We have that $H_{2m+1} = \mathsf{Obf}(\mathsf{Merge}(BP_1, BP_1, 1))$. We switch back so that the first program is active (which in this case is the same as the second program BP_1)

$$H_{2m+2} = \mathsf{Obf}(\mathsf{Merge}(BP_1, BP_1, 0))$$

- For $i = 1, 2 \dots m$, let
$$H_{2m+i+2} = \mathsf{Obf}(\mathsf{Merge}(BP_1, \mathsf{Hyb}_i(I, BP_1), 0))$$
We change the second program Merge takes as input from BP_1 to I, one level at a time as before. Finally we get
$$H_{3m+2} = i\mathcal{O}(C_n^1) = \mathsf{Obf}(\mathsf{Merge}(BP_1, I, 0))$$
which is the obfuscation of the circuit C_n^1.

Recall that by assumption D distinguishes between $\{i\mathcal{O}(C_n^0)\}_{n\in\mathbb{N}}$ and $\{i\mathcal{O}(C_n^1)\}_{n\in\mathbb{N}}$. That is, there is a polynomial p such that for infinitely many n

$$|Pr[D(H_0) = 1] - Pr[D(H_{3m+2})]| > 1/p(n)$$

By the above hybrid argument, D must distinguish between a pair of consecutive hybrids. That is, there exists some $i \in \{0, 1, \dots 3m + 1\}$ such that

$$|Pr[D(H_i) = 1] - Pr[D(H_{i+1})]| > 1/4mp(n)$$

We now show that H_i and H_{i+1} can be expressed as the $\mathsf{Obf}(BP)$ and $\mathsf{Obf}(BP')$ respectively where BP and BP' are (widened) branching programs that differ in only two levels and agree on all inputs. Furthermore, both BP and BP' have the property that for all inputs x the first column of the output matrix $\mathsf{col}_1(\mathsf{P_{out}}(x))$ is the same for BP and BP', and depends only on the output of these programs on x. More formally,

Claim 14. *There exist branching programs BP and BP' of length $m' = m + 2$ and width 10 such that*

- *$H_i = \mathsf{Obf}(BP)$ and $H_{i+1} = \mathsf{Obf}(BP')$.*
- *BP and BP' differ in at most 2 levels.*
- *For all $x \in \{0, 1\}^n$, $BP(x) = BP'(x)$.*
- *Let ${\mathsf{P_{out}}}^{BP}(\cdot)$ and ${\mathsf{P_{out}}}^{BP'}(\cdot)$ be the functions computing the output matrices for BP and BP' respectively. There exist length 10 vectors v_0 and v_1 such that for every $x \in \{0, 1\}^n$, $\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{col}_1({\mathsf{P_{out}}}^{BP'}(x)) = v_{BP(x)}$*

Proof. Let $v_1 = \mathsf{extend}(\mathsf{col}_1(\mathsf{P_{accept}}))$ and $v_0 = \mathsf{extend}(\mathsf{col}_1(\mathsf{P_{reject}}))$ where $\mathsf{P_{accept}}$ and $\mathsf{P_{reject}}$ are the accepting and rejecting matrices from Theorem 4 for branching programs of input lengths n, and extend extends a length w vector by appending w zeroes. We consider three cases: when i is equal to m, $2m+1$ and otherwise.

Case 1: $i = m$**:** By definition of H_i and H_{i+1}, the branching programs BP and BP' are $\mathsf{Merge}(BP_0, BP_1, 0)$ and $\mathsf{Merge}(BP_0, BP_1, 1)$ respectively. By Claim 8, BP and BP' differ in the "switch" matrices, which make up the first and last level. Furthermore, BP and BP' compute BP_0 and BP_1 respectively which are equivalent programs by assumption. It remains to show the fourth condition. By Claim 8, the first column of the output matrix for a widened branching program only depends on the first column of the output matrix of the active program. Hence, for every input x, $\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{extend}(\mathsf{col}_1({\mathsf{P_{out}}}^{BP_0}(x)))$.

By Theorem 4, ${\mathsf{P_{out}}}^{BP_0}(x)$ is either $\mathsf{P_{accept}}$ or $\mathsf{P_{reject}}$ depending on the output $BP_0(x)$. Therefore, for all inputs x such that $BP(x) = 0$,

$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{extend}(\mathsf{col}_1(\mathsf{P_{reject}})) = v_0$$

Similarly, for all inputs x such that $BP(x) = 1$,

$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{extend}(\mathsf{col}_1(\mathsf{P_{accept}})) = v_1$$

The same argument holds for BP' too, in which case BP_1 is active and has the same accepting and rejecting permutations $\mathsf{P_{accept}}$ and $\mathsf{P_{reject}}$ by Theorem 4. Therefore, for all inputs x,

$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP'}(x)) = v_{BP_1(x)}$$

Since $BP_0(x) = BP_1(x) = BP(x)$ for all x, the claim follows.

Case 2: $i = 2m+1$**:** By definition of H_i and H_{i+1}, the branching programs BP and BP' are $\mathsf{Merge}(BP_1, BP_1, 0)$ and $\mathsf{Merge}(BP_1, BP_1, 1)$ respectively. As before, these programs differ in the first and level only. Furthermore, both BP and BP' compute the same function, as the active program is the same (BP_1). Also, directly from Claim 8 and Theorem 4 we have that for all inputs $x \in \{0,1\}^n$,

$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{col}_1({\mathsf{P_{out}}}^{BP'}(x)) = \mathsf{extend}(\mathsf{col}_1({\mathsf{P_{out}}}^{BP_1}(x))) = v_{BP_1(x)} = v_{BP(x)}$$

Case 3: $i \neq m$ **and** $i \neq 2m+1$**:** First, consider the subcase when $i < m$ or $i > 2m+1$. The programs BP and BP' are of the form $\mathsf{Merge}(BP_0, P_i)$ and $\mathsf{Merge}(BP_0, P_{i+1})$ respectively where P_i and P_{i+1} are branching programs that differ only in the $i+1^{th}$ level. By Claim 8, BP and BP' differ only in the $i+1^{th}$ level too. Furthermore, in both BP and BP', the active program is BP_0. Hence BP and BP' compute the same function and similarly as the previous case, we have that for all inputs $x \in \{0,1\}^n$,

$$\mathsf{col}_1({\mathsf{P_{out}}}^{BP}(x)) = \mathsf{col}_1({\mathsf{P_{out}}}^{BP'}(x)) = \mathsf{extend}(\mathsf{col}_1({\mathsf{P_{out}}}^{BP_0}(x))) = v_{BP_0(x)} = v_{BP(x)}$$

The case when $m < i < 2m+1$ follows similarly. This concludes the proof of the claim. □

This concludes the first part of the proof. At this point we have that there is a polynomial p such that for infinitely many n there exist branching programs BP and BP' with the properties described in Claim 14 such that

$$|Pr[D(\mathsf{Obf}(BP)) = 1] - Pr[D(\mathsf{Obf}(BP'))]| > 1/4mp(n)$$

In the next part we show that the distinguisher D can be used to break the semantic security game of the graded encoding scheme used by Obf.

5.2 Applying Semantic Security

Fix $n \in \mathbb{N}$, and let $BP = \{\mathsf{inp}(i), B_{i,0}, B_{i,1}\}_{i\in[m']}$ and $BP' = \{\mathsf{inp}(i), B'_{i,0}, B'_{i,1}\}_{i\in[m']}$. Let $l_1, l_2 \in [m]$ be the levels in which BP and BP' differ. All other matrices of BP and BP' are the same. That is, for every $i \notin \{l_1, l_2\}, b \in \{0,1\}$ we have that $B_{i,b} = B'_{i,b}$.

Define nuPPT M that gets BP and BP' as non-uniform advice and on input the public parameters pp that describe a $(k, \mathbb{Z}_p)$-graded encoding scheme samples m' random invertible 10×10 matrices over $\mathbb{Z}_p$, $\{R_i\}_{i\in[m']}$ and $2m'$ random scalars from $\mathbb{Z}_p$, $\{\alpha_{i,b}\}_{i\in[m'],b\in\{0,1\}}$. M then uses these matrices and scalars to randomize BP and BP' as described by $\mathsf{Rand}(\cdot, p)$ to obtain $\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m'],b\in\{0,1\}}$, $\{\alpha_{i,b} \cdot \tilde{B}'_{i,b}\}_{i\in[m'],b\in\{0,1\}}$ and $\mathbf{t}$. M outputs

$$\vec{m_0} = (\{\alpha_{l_1,b} \cdot \tilde{B}_{l_1,b}\}_{b\in\{0,1\}}, \{\alpha_{l_2,b} \cdot \tilde{B}_{l_2,b}\}_{b\in\{0,1\}})$$

$$\vec{m_1} = (\{\alpha_{l_1,b} \cdot \tilde{B'}_{l_1,b}\}_{b\in\{0,1\}}, \{\alpha_{l_2,b} \cdot \tilde{B'}_{l_2,b}\}_{b\in\{0,1\}})$$
$$\vec{z} = (\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m']/\{l_1,l_2\},b\in\{0,1\}}, \mathbf{t})$$

We observe that $D(\mathsf{Obf}(BP))$ (resp. $D(\mathsf{Obf}(BP')))$ is simply the output of D when playing the semantic security game with the message sampler M and parameterized by the bit $b = 0$ (resp. $b = 1$). Formally, there exist polynomials $q = O(m), k = O(m)$, constant c and set ensembles $\{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}}$ such that for all $n \in \mathbb{N}$

$$D(\mathsf{Obf}(BP)) \equiv \mathbf{Output}_0(q, k, c, D, M, n, (\vec{S}_n, \vec{T}_n))$$

and

$$D(\mathsf{Obf}(BP')) \equiv \mathbf{Output}_1(q, k, c, D, M, n, (\vec{S}_n, \vec{T}_n))$$

where $\vec{S}_n$, $\vec{T}_n$ contain sets from $\mathsf{SetSystem}(m', n, \mathsf{inp})$ and $\mathbf{Output}_b$ is as defined in Definition 9.

To see this, observe that $\vec{m_0}$ and $\vec{m_1}$ consist of a constant number of ring elements while $\vec{z}$ contains polynomially many ring elements. Note that the distribution of $(\vec{m_0}, \vec{z})$ is identical to $\mathsf{Rand}(BP, p)$ and the distribution of $(\vec{m_1}, \vec{z})$ is identical to $\mathsf{Rand}(BP', p)$. When these elements are encoded under sets in $\mathsf{SetSystem}(m', n, \mathsf{inp})$[23] then we obtain the distributions $\mathsf{Obf}(BP)$ and $\mathsf{Obf}(BP')$ respectively.

Recall that for infinitely many n,

$$|Pr[D(\mathsf{Obf}(BP)) = 1] - Pr[D(\mathsf{Obf}(BP'))]| > 1/4mp(n)$$

Since the graded encoding scheme is semantically secure, it must be that M is not a $(q, k, c, \{(\vec{S}_n, \vec{T}_n)\}_{n\in\mathbb{N}})$-respecting message sampler. Therefore, there exists a polynomial p' such that for infinitely many $n \in \mathbb{N}$, for some $(\mathsf{sp}, \mathsf{pp}), \in \mathsf{InstGen}(1^n, 1^{k(n)})$ and some algebraic adversary A,

$$Pr_{A,\mathsf{GObf}}[A^{\mathcal{M}(\mathsf{GObf}(BP,\mathsf{pp}))}(1^n) \neq A^{\mathcal{M}(\mathsf{GObf}(BP',\mathsf{pp}))}(1^n)] > 1/p'(n)$$

In the remainder of the proof we show that if BP and BP' agree on all inputs then such an algebraic adversary A cannot exist. Similar statements were shown in [BGK+13] and [BR14]. In particular, GObf is a simplified version of the obfuscator of [BGK+13], which [BGK+13] shows is VBB secure against algebraic adversaries. We will follow the structure of the proof in [BGK+13], but cannot use it in a black-box way due to the differences in the construction and the fact that their proof only works for branching programs that have unique accepting and rejecting output matrices. The branching programs we consider may not have this property.

5.3 Simulating an Algebraic Adversary

We show the following claim.

Claim 15. *There exists Turing machine* Sim *such that for every nuPPT A there exists a negligible function ε such that the following holds. For every $n \in \mathbb{N}$, every length $m = poly(n)$, for $k \in$* $\mathsf{SetSystem}(1^n, 1^m)$*, for every* $(\mathsf{sp}, \mathsf{pp}) \in \mathsf{InstGen}(1^n, 1^k)$*, for every width 10 branching program BP with labeling function* $\mathsf{inp} : [m] \to [n]$ *and for which there exist $v_0, v_1 \in \{0,1\}^{10}$ such that on every input $x \in \{0,1\}^n$,* $\mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x)) = v_{BP(x)}$*, it holds that*

$$Pr_{A,\mathsf{GObf},\mathsf{Sim}}[(A^{\mathcal{M}(\mathsf{GObf}(BP,\mathsf{pp}))}(1^n) \neq \mathsf{Sim}^{BP}(1^n, 1^m, A, \mathsf{pp}, \mathsf{inp}, v_0, v_1))] \leq \varepsilon(n)$$

[23]Every element is encoded with the corresponding set in $\mathsf{SetSystem}(m', n, \mathsf{inp})$. For example, elements from $\alpha_{i,b}\tilde{B}_{i,b}$ are encoded under the set $S_{i,b}$ in $\mathsf{SetSystem}(m', n, \mathsf{inp})$

Sim's strategy will be to run A and simulate the oracle $\mathcal{M}(\mathsf{GObf}(BP, \mathsf{pp}))$ for A. Recall that $\mathsf{GObf}(BP, \mathsf{pp})$ contains a list of the ring elements in $\mathsf{Rand}(BP)$ paired with the corresponding set in $\vec{S} = \mathsf{SetSystem}(n, m, \mathsf{inp})$. $\mathcal{M}(\mathsf{GObf}(BP, \mathsf{pp}))$ when queried with an arithmetic circuit C, first checks if C is $\vec{S}$-respecting and then outputs the result of C on the ring elements in $\mathsf{Rand}(BP)$.

Sim only has oracle access to BP and can not run $\mathcal{M}$ directly on $\mathsf{GObf}(BP, \mathsf{pp})$. However, we show in the following lemma that Sim can simulate the output of $\mathcal{M}$ on a single query. In particular, except with negligible probability (over $\mathsf{GObf}(BP, \mathsf{pp})$ and the simulation) the simulated query response will be *identical* to the actual query response. Furthermore, this response depends only on the queried arithmetic circuit C and the input-output behavior of BP, and is thus independent of $\mathsf{GObf}(BP, \mathsf{pp})$, even in the real experiment. Since A makes only polynomially many queries, by the Union Bound, it follows that except with negligible probability Sim succeeds in correctly simulating all queries. Therefore, it suffices to show the following lemma.

Lemma 16. *There exists a Turing machine* CSim *such that for every* $m, n, w \in \mathbb{N}$, $v_0, v_1 \in \{0,1\}^w$, *labeling function* $\mathsf{inp} : [m] \to [n]$, *prime number* p, *and* $\vec{S}$*-respecting arithmetic circuit* C *where* $\vec{S} = \mathsf{SetSystem}(m, n, \mathsf{inp})$, *the following holds. For every branching program* BP *of length* m, *width* w *and labeling function* inp *for which on every input* x, $\mathsf{col}_1(\mathsf{P}_{\mathsf{out}}(x)) = v_{BP(x)}$ *it holds that*

$$Pr[\mathsf{isZero}(C(\mathsf{Rand}(BP, p))) \neq \mathsf{CSim}^{BP}(1^m, p, C, v_0, v_1)] \leq 32wm/p$$

The proof of the lemma follows the structure of the VBB simulation in [BGK+13], appropriately adapted to deal with the fact that our branching programs do not have a unique output by relying on Theorem 9.

Proof. Roughly speaking the lemma follows from the the property that $\vec{S}$-respecting arithmetic circuits, due to the straddling set systems in $\vec{S}$, can only evaluate expressions that are "consistent" with some inputs. In particular, following [BGK+13], the polynomial C evaluates can be expressed as the sum of *single-input terms* where each *single-input term* is a function of those elements of that are consistent with some input to the branching program. Next, we rely on Theorem 9 to show that the sum of these single-input terms will depend only on the value of the branching program on these inputs.

The following proposition states that the function a $\vec{S}$-respecting arithmetic circuit computes can be expressed as the sum of several *single-input* terms. This decomposition is very similar to the one shown in [BGK+13].[24]

Proposition 1. *Fix* $m, n, w \in \mathbb{N}$ *and* $\mathsf{inp} : [m] \to [n]$. *Let* $\vec{S} = \mathsf{SetSystem}(m, n, \mathsf{inp}) = (\{S_{i,b}\}_{i \in [m], b \in \{0,1\}}, S_t)$, *and let* C *be any* $\vec{S}$*-respecting arithmetic circuit. There exists a set* $X \subseteq \{0,1\}^n$ *of inputs such that*

(i)

$$C(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) \equiv \sum_{x \in X} C_x(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t})$$

where each C_x *is a* $\vec{S}$*-respecting arithmetic circuit, whose input wires are labelled only with sets respecting a single input* $x \in \{0,1\}^n$, *that is, only with sets* $\in \{S_{i,x[\mathsf{inp}(i)]}\}_{i \in [m]} \cup \{S_t\}$.

(ii) *For each* C_x *above, for every branching program* BP *of width* w *and length* m *on* n *input bits, with input labelling function* inp, *every prime* p, *and every* $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i \in m, b \in \{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP, p)$

$$C_x(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i \in [m], b \in \{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\mathsf{inp}(i)]}\}_{i \in [m]}, \mathbf{t})$$

where p_x *is some polynomial, and* $\alpha_x = (\prod_{i \in [m]} \alpha_{i,x[\mathsf{inp}(i)]})$. *Furthermore, when* p_x *is viewed as a sum of monomials, each monomial contains exactly one entry from each* $\tilde{B}_{i,x[\mathsf{inp}(i)]}$, *and one entry from* $\mathbf{t}$.

[24] The key difference is that [BGK+13] proves such a decomposition for "dual-input" branching program, and use the "dual-input" property to show that there are only polynomially many terms in the decomposition.

The proof of Proposition 1 uses the following lemma:

Lemma 17. *Fix $m, n, w \in \mathbb{N}$ and $\mathsf{inp} : [m] \to [n]$. Let $\vec{S} = \mathsf{SetSystem}(m, n, \mathsf{inp}) = (\{S_{i,b}\}_{i\in[m],b\in\{0,1\}}, S_t)$, and let C be any weakly $\vec{S}$-respecting arithmetic circuit whose output wire is tagged with $T \subseteq [k]$. Then there exists a set $U \subseteq \{0, 1, *\}^m$ such that for every branching program BP of width w and length m on n input bits, with input tagging function inp, every prime p, and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in m,b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP, p)$,*

(i)

$$C(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) \equiv \sum_{u\in U} C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$$

where each C_u is a weakly $\vec{S}$-respecting arithmetic circuit, whose input wires are tagged only with sets $\in \{S_{i,u[i]}\}_{i\in[m]:u[i]\neq*} \cup \{S_t\}$, and whose output wire is tagged with T.

(ii) *Each C_u above is the sum of several "monomial" circuits, where each monomial circuit performs only multiplications of elements in $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in m,b\in\{0,1\}}, \mathbf{t})$, is weakly $\vec{S}$-respecting, and has output wire tagged with T.*

(iii) *For each C_u above,*

$$C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i\in[m]:u[i]\neq*}, \mathbf{t})$$

where p_u is some polynomial, and $\alpha_u = (\prod_{i\in[m]:u[i]\neq*} \alpha_{i,u[i]})$. Furthermore, when p_u is viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,u[i]}$ such that $u[i] \neq *$, and possibly one entry from $\mathbf{t}$. Further, p_u can be computed by a weakly $\vec{S}$-respecting circuit whose output wire is tagged with T.

The lemma can be proved using a simple induction. We provide a complete proof of the lemma in Appendix B. Given this lemma, the proof of Proposition 1 is as follows:

Proof. **Part (i)** We consider the special case of Lemma 17 part (i), in which C is $\vec{S}$-respecting (as opposed to only weakly $\vec{S}$-respecting). In this case, we have that each C_u in the decomposition of C is also $\vec{S}$-respecting, and in particular, each C_u for $u \in U$ has its output wire tagged with the universe set $[k]$.

We first observe that for any C_u in the decomposition of C, u cannot contain $*$. This is because the output of C_u is tagged with $[k]$, and thus must have at least one input wire tagged with either of $S_{i,0}$ or $S_{i,1}$ for each i, or else the straddling set $\mathbb{S}^{\mathsf{inp}(i)}$ will be incomplete, and thus the output wire cannot be tagged with $[k]$.

Further, we observe that for every $u \in U$, for every $j \in [n]$, there must be a bit $b_j \in \{0, 1\}$ such that for every $i \in [m]$ such that $\mathsf{inp}(i) = j$, $u[i] = b_j$. This can be seen by considering any monomial circuit in C_u individually. Recall from Lemma 17 part (ii) that C_u is formed by summing some number of monomials circuits, each of which is $\vec{S}$-respecting and has output wire tagged with $[k]$. This means that $\mathbb{S}^j \subseteq [k]$ is covered by the elements of the monomial. However, since $\mathbb{S}^j$ is constructed as a straddling set, the only way to cover $\mathbb{S}^j$ in a monomial circuit that only contains multiplication gates, is by using either all sets from $\{S_{i,0} : \mathsf{inp}(i) = j\}_{i\in m}$ or all sets from $\{S_{i,1} : \mathsf{inp}(i) = j\}_{i\in m}$. This means, correspondingly, that u must be such that there is a bit $b_j \in \{0, 1\}$, for every $i \in [m]$ such that $\mathsf{inp}(i) = j$, $u[i] = b$. Define $x \in \{0, 1\}^n$ so that $x[j] = b_j$ for all $j \in [n]$. In this way, we can define a one-to-one correspondence from each $u \in U$ to corresponding $x \in \{0, 1\}^n$, and we simply relabel each C_u to the corresponding C_x to get the desired decomposition of C. We observe that the additional conditions on each C_x can be achieved from the corresponding conditions on C_u as guaranteed by Lemma 17.

Part (ii) Part (ii) follows directly from Part (i) of this proposition, together with Lemma 17 part (iii), and the observation that each C_u in Lemma 17 is relabelled to C_x for some $x \in \{0,1\}^n$ in Part (i) of this proposition. □

Now we are ready to describe the simulator CSim. CSim gets as input 1^m, prime p, a $\vec{S}$-respecting circuit C, vectors v_0, v_1 and has oracle access to a length m branching program BP. Let X be the set of inputs and $\{p_x\}_{x\in X}$ be the single-input polynomials corresponding to the decomposition of C. For every $x \in X$, CSim queries BP on x, samples $d_x \leftarrow \mathsf{KSim}(1^m, p, v_{BP(x)})$ and checks whether $p_x(d_x) = 0$. CSim outputs 1 if and only if for every input $x \in X$, $p_x(d_x) = 0$.

Now we prove correctness of our simulation. First, we prove some claims that will be useful. In each of these claims, let proj_x be defined with respect to the labeling function inp of the branching program BP. The following claim states that if $C(\mathsf{Rand}(BP,p))$ is always zero, then every single-input term is always zero.

Claim 18. *If $Pr[C(\mathsf{Rand}(BP,p) = 0] = 1$ then for every input $x \in X$,*

$$Pr[p_x(\mathsf{proj}_x(\mathsf{Rand}^B(BP,p))) = 0] = 1$$

Proof. Consider a fixed $d = (\{\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$ in the support of $\mathsf{Rand}^B(BP,p)$ and let $C_d(\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}) = C(\{\alpha_{i,b}\cdot\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$. By Proposition 1, we know that

$$C_d(\{\alpha_{i,b}\}) = \sum_{x\in X}(\prod_{i\in[m]} \alpha_{i,x[\mathsf{inp}(i)]})p_x(\mathsf{proj}_x(d))$$

and C_d is a degree $m+2$ polynomial. By assumption, $C(\mathsf{Rand}(BP,p))$ is always zero (over the support of $\mathsf{Rand}(BP,p)$); hence, $C_d(\{\alpha_{i,b}\}) = 0$ for all non-zero $\{\alpha_{i,b}\}$. By the Schwartz-Zippel lemma, this can happen only if C_d is the zero polynomial. By the structure of C_d, this implies that for every $x \in X$, $p_x(\mathsf{proj}_x(d)) = 0$. This argument works for every fixed value of d, hence we have that for every $x \in X$, $\Pr[p_x(\mathsf{proj}_x(\mathsf{Rand}^B(BP,p))) = 0] = 1$. □

The next claim states that if $C(\mathsf{Rand}(BP,p))$ is not always zero, then it is zero with small probability. Furthermore, there exists a single-input term that is zero with small probability.

Claim 19. *For any $\vec{S}$-respecting circuit C, if $Pr[C(\mathsf{Rand}(BP,p)) = 0] < 1$ then the following holds.*

1. *$Pr[C(\mathsf{Rand}(BP,p)) = 0] \le 16wm/p$*
2. *There exists $x \in X$ such that $Pr[p_x(\mathsf{proj}_x(\mathsf{Rand}^B(BP,p))) = 0] \le 16wm/p$, where X is obtained from the decomposition of C by Proposition 1.*

Proof. We start by showing part 1.

Part 1: If $\mathsf{Rand}(BP,p) = \mathsf{Rand}^\alpha(\mathsf{Rand}^B(BP,p))$ can be expressed as a low-degree ($\le 2w$) polynomial on uniformly random values in $\mathbb{Z}_p$—namely, the α's and the randomization matrices R_i's—then by the Schwartz-Zippel lemma the first part of the claim directly follows. However, there are two barriers to applying this argument:

- Rand^B does not sample uniformly random matrices $\{R_i\}_{i\in[m]}$; rather, it chooses uniformly random *invertible* matrices R_i. Similarly, Rand^α does not sample uniformly random $\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}$; rather, it chooses uniformly random *non-zero* $\alpha_{i,b}$.
- Rand^B also needs to compute inverses R_i^{-1} to R_i for every $i \in [m]$ (which may no longer be expressed as low degree polynomials in the matrices $\{R_i\}_{i\in[m]}$).

To handle the second issue, consider the distribution $\mathsf{Rand}^{B}_{adj}(BP,p)$ that is defined exactly as $\mathsf{Rand}^{B}(BP,p)$ except that for every $i \in [m]$ it uses $adj(R_i) = R_i^{-1}det(R_i)$ instead of R_i^{-1}. Note that every entry of the adjoint of a $w \times w$ matrix M is some cofactor of M (obtained by the determinant of the $w-1 \times w-1$ matrix obtained by deleting some row and column of A). Hence every entry of $adj(R_i)$ can be expressed as a degree w polynomial in R_i. Let $\mathsf{Rand}_{adj}(BP,p) = \mathsf{Rand}^{\alpha}(\mathsf{Rand}^{B}_{adj}(BP,p))$. It follows that $\mathsf{Rand}_{adj}(BP,p)$ is computed by degree (at most) $2w$ polynomial in the matrices $\{R_i\}_{i\in[m]}$ and scalars $\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}$.

Furthermore, we show that $Pr[C(\mathsf{Rand}_{adj}(BP,p)) = 0] = Pr[C(\mathsf{Rand}(BP,p)) = 0]$. Recall that by Proposition 1,

$$C \equiv \sum_{x\in X} C_x$$

and for each C_x above and every $(\{\alpha_{i,b}\cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP,p)$,

$$C_x(\{\alpha_{i,b}\cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\mathsf{inp}(i)]}\}_{i\in[m]}, \mathbf{t})$$

where $\alpha_x = (\prod_{i\in[m]} \alpha_{i,x[\mathsf{inp}(i)]})$ and p_x is a polynomial such that, when viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,x[\mathsf{inp}(i)]}$, and one entry from $\mathbf{t}$. Recall that for every $i \in [m]$,

$$\tilde{B}_{i,x[\mathsf{inp}(i)]} = R_{i-1}B_{i,x[\mathsf{inp}(i)]}R_i^{-1}$$

For every $i \in [m]$, replacing R_i^{-1} with $adj(R_i)$ has the effect of multiplying each monomial in p_x with the scalar $det(R_i)$. Hence

$$C_x(\mathsf{Rand}_{adj}(BP,p)) = (\prod_{i\in[m]} det(R_i)) \cdot C_x(\mathsf{Rand}(BP,p))$$

Since C is the sum of such C_x terms, it holds that $C(\mathsf{Rand}_{adj}(BP,p)) = (\prod_{i\in[m]} det(R_i))C(\mathsf{Rand}(BP,p))$. For every $i \in [m]$, by invertibility, $det(R_i) \neq 0$ and hence

$$Pr[C(\mathsf{Rand}_{adj}(BP,p)) = 0] = Pr[C(\mathsf{Rand}(BP,p)) = 0]$$

So far, we have that $\mathsf{Rand}_{adj}(BP,p)$ is computed by a degree $2w$ polynomial in the matrices $\{R_i\}_{i\in[m]}$ and scalars $\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}$. However the first issue remains: each R_i is uniformly random invertible and each $\alpha_{i,b}$ is uniformly random non-zero, whereas we need them to be uniformly random. Consider the distribution $\mathsf{Rand}_{adj,U}(BP,p)$ that is obtained by the computing the same polynomial on uniformly random matrices $\{R_i\}_{i\in[m]}$ and scalars $\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}$ over $\mathbb{Z}_p$. In Claim 24, we show that the statistical distance between $\mathsf{Rand}_{adj}(BP,p)$ and $\mathsf{Rand}_{adj,U}(BP,p)$ is at most $8wm/p$. Furthermore, the support of $\mathsf{Rand}_{adj,U}(BP,p)$ contains the support of $\mathsf{Rand}_{adj}(BP,p)$. This implies that if $Pr[C(\mathsf{Rand}_{adj}(BP,p)) = 0] < 1$ then $Pr[C(\mathsf{Rand}_{adj,U}(BP,p)) = 0] < 1$.

We now turn to proving the statement of the claim. Using facts shown above, we have that

$$Pr[C(\mathsf{Rand}(BP,p)) = 0] < 1 \implies Pr[C(\mathsf{Rand}_{adj}(BP,p)) = 0] < 1 \implies Pr[C(\mathsf{Rand}_{adj,U}(BP,p)) = 0] < 1$$

By Proposition 1, C evaluates a $m+1$ degree polynomial, and $\mathsf{Rand}_{adj,U}(BP,p)$ is computed by a degree $2w$ polynomial in uniformly random values in $\mathbb{Z}_p$. By the Schwartz-Zippel lemma,

$$Pr[C(\mathsf{Rand}_{adj,U}(BP,p)) = 0] < 1 \implies Pr[C(\mathsf{Rand}_{adj,U}(BP,p) = 0] \leq 2w(m+1)/p \leq 8wm/p$$

We have that the statistical distance between $\mathsf{Rand}_{adj,U}(BP,p)$ and $\mathsf{Rand}_{adj}(BP,p)$ is at most $8wm/p$. Therefore, $Pr[C(\mathsf{Rand}(BP,p)) = 0] = Pr[C(\mathsf{Rand}_{adj}(BP,p)) = 0] \leq 16wm/p$ thus proving the first part of the claim. We proceed to show part 2.

Part 2: By Proposition 1, for every $x \in X$, there exists a $\vec{S}$-respecting arithmetic circuit C_x such that for every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP, p)$,

$$C_x(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = \alpha_x \cdot p_x(\{\tilde{B}_{i,x[\mathsf{inp}(i)]}\}_{i\in[m]}, \mathbf{t})$$

where $\alpha_x = (\prod_{i\in[m]} \alpha_{i,x[\mathsf{inp}(i)]})$ and $C = \sum_{x\in X} C_x$. In particular, $p_x(\{\tilde{B}_{i,x[\mathsf{inp}(i)]}\}_{i\in[m]}, \mathbf{t}) = 0$ iff $C_x(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = 0$ (since $\alpha_{i,b}$ is non-zero).

Thus, we have that

$$Pr[C(\mathsf{Rand}(BP,p))) = 0] = Pr[C_x(\mathsf{Rand}^{\alpha}(\mathsf{Rand}^{B}(BP,p))) = 0] = Pr[p_x(\mathsf{proj}_x(\mathsf{Rand}^{B}(BP,p))) = 0]$$

There must exist an input $x \in X$ such that $Pr[C_x(\mathsf{Rand}(BP,p))) = 0] < 1$ or else $Pr[C(\mathsf{Rand}(BP,p))) = 0] = 1$. By the first part of the claim, it follows that

$$Pr[C(\mathsf{Rand}(BP,p))) = 0] \leq 16wm/p,$$

which concludes the proof. □

Now we analyze the correctness of the simulator CSim. We consider the following two cases: when $C(\mathsf{Rand}(BP,p))$ is always zero, and otherwise.

Case 1: $Pr[C(\mathsf{Rand}(BP,p)) = 0] = 1$**:** In this case we will show that the simulation always succeeds. If $Pr[C(\mathsf{Rand}(BP,p)) = 0] = 1$ then by Claim 18, for every $x \in X$, $Pr[p_x(\mathsf{proj}_x(\mathsf{Rand}^{B}(BP,p))) = 0] = 1$. Recall that $\mathsf{KSim}(1^m, p, v_{BP(x)})$ simulates $\mathsf{proj}_x(\mathsf{Rand}^{B}(BP,p))$ perfectly. Therefore, CSim always outputs 1 and hence succeeds.

Case 2: $Pr[C(\mathsf{Rand}(BP,p)) = 0] < 1$**:** In this case, by the first part of Claim 19 we have that

$$Pr[\mathsf{isZero}(C(\mathsf{Rand}(BP,p))) = 1] \leq 16wm/p$$

By the perfect simulation of KSim, we have that

$$Pr[\mathsf{CSim}^{BP} = 1] = Pr[\forall x\ (d_x \leftarrow \mathsf{proj}_x(\mathsf{Rand}^{B}(BP,p)) : p_x(d_x) = 0)]$$

By second part of Claim 19 there exists input x_C such that $Pr[p_{x_C}(\mathsf{proj}_{x_C}(\mathsf{Rand}^{B}(BP,p))) = 0] \leq 16wm/p$. Therefore,

$$Pr[\mathsf{CSim}^{BP} = 1] \leq Pr[p_{x_C}(\mathsf{proj}_{x_C}(\mathsf{Rand}^{B}(BP,p))) = 0] \leq 16wm/p$$

Therefore, by a union bound we have that

$$Pr[\mathsf{isZero}(C(\mathcal{D})) = \mathsf{CSim}^{BP} = 0] > 1 - 32wm/p$$

This concludes the proof of the lemma. □

□

5.4 Restricting to Entropic Message Samplers

We here show that the message samper M satisfies the required high-entropy condition (required by the notion of entropic semantical security); that is, M is entropically valid.

Recall that the message sampler M in the proof of Theorem 13 gets as input the description of a ring $R = \mathbb{Z}_p$ and samples $(\vec{m}_0, \vec{m}_1, \vec{z})$ such that $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$ are the "randomizations" (as defined in the description of Rand) of fixed branching programs. We now show the following proposition, which combined with the fact that the length m of the branching programs is polynomial in $\log|R|$ (recall that $R = \mathbb{Z}_p$ where p is a prime exponential in the multilinearity parameter k which is $< 3m$), implies that the output of a non-terminal set-respecting circuit on input $(\vec{m}_b, \vec{z})$ (for both $b \in \{0,1\}$) has min-entropy $\log|R| - O(\log\log|R|)$, as required.

Proposition 2. *Let BP be a branching program of length m, width w, input length n and input labeling function* inp. *Let p be a prime and $\vec{S} = \mathsf{SetSystem}(m, n, \mathsf{inp})$. Let C be a non-terminal $\vec{S}$-respecting arithmetic circuit that a non-zero polynomial. Then we have that*

$$H_\infty(C(\mathsf{Rand}(BP, p))) \geq \log(\frac{p}{12wm})$$

or equivalently, for any fixed output $a \in \mathbb{Z}_p$

$$Pr[C(\mathsf{Rand}(BP, p)) = a] \leq 12wm/p$$

Proof. Let T be the set that tags the output wire of C as per the construction given in Definition 6. Since C is non-terminal $\vec{S}$-respecting, we have that T is a strict subset of $[k]$ where $(k, \vec{S}) = \mathsf{SetSystem}(m, n, \mathsf{inp})$. By Lemma 17 part (iii), there exists a set U of labels $u \in \{0, 1, *\}$ such that for every $(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j\in[m],b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP, p)$ we have that

$$C(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j\in[m],b\in\{0,1\}}, \mathbf{t}) = \sum_{u\in U} \alpha_u \cdot p_u(\{\tilde{B}_{j,u[j]}\}_{j\in[m]:u[j]\neq *}, \mathbf{t}) \tag{4}$$

where $\alpha_u = \prod_{j\in[m]:u[j]\neq *} \alpha_{j,u[j]}$. Furthermore, each p_u is computed by a weakly $\vec{S}$-respecting circuit whose output wire is also tagged with T. Since C computes a non-zero polynomial, there must exist $v \in U$ such that p_v is a non-zero polynomial. We now have the following claim.

Claim 20. *$Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j\in[m]:v[j]\neq *}, \mathbf{t}) = 0] \leq 10wm/p$.*

Proof. To see this, we first observe that since T is a strict subset of $[k]$ and p_v is computed by a $\vec{S}$-respecting circuit whose output wire is tagged with T, either p_v does not operate on some level of the branching program or it does not operate on $\mathbf{t}$; that is, either,

- there exists $j \in [m]$ such that $v[j] = *$, or
- p_v is not a function of $\mathbf{t}$.

In the first case, by an argument similar to that in Claim 11, we can show that the distribution $(\{\tilde{B}_{j,v[j]}\}_{j\in[m]:v[j]\neq *}, \mathbf{t})$ is identical to the distribution $(\{R_j\}_{j\in[m]:v[j]\neq *}, \mathsf{col}_1(R_{m+1}))$ where $\{R_j\}_{j=1}^{m+1}$ are random invertible matrices over $\mathbb{Z}_p^{w\times w}$. By Claim 24, this distribution is statistically $8wm/p$-close to the distribution where each matrix entry is uniformly random in $\mathbb{Z}_p$. Furthermore, since p_v is computed by a $\vec{S}$-respecting circuit, it is of degree at most $m + 1 < 2wm$. By the Schwartz Zippel lemma, the evaluation of p_v on such random inputs from $\mathbb{Z}_p$ is zero with probability at most $2wm/p$. All in all, we have $Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j\in[m]:v[j]\neq *}, \mathbf{t}) = 0] \leq 10wm/p$.

In the second case, p_v acts on the $\{\tilde{B}_{j,v[j]}\}_{j\in[m]}$. Following Claim 11, this distribution is identical to that of m random invertible matrices over $\mathbb{Z}_p^{w\times w}$. Similarly to the first case, it follows that $Pr[p_v(\{\tilde{B}_{j,v[j]}\}_{j\in[m]}) = 0] \leq 10wm/p$. □

Let E be the event that $p_v(\{\tilde{B}_{j,v[j]}\}_{j\in[m]:v[j]\neq *}, \mathbf{t}) \neq 0$. For any fixed output $a \in \mathbb{Z}_p$ we have that

$$Pr[C(\mathsf{Rand}(BP, p)) = a] \leq Pr[C(\mathsf{Rand}(BP, p)) = a|E] + Pr[\bar{E}] \tag{5}$$

For a fixed $\{\tilde{B}_{j,b}\}_{j\in[m],b\in\{0,1\}}$ let $q_{(\tilde{B},a)}$ be a polynomial in variables $\{\alpha_{j,b}\}_{j\in[m],b\in\{0,1\}}$ such that

$$q_{(\tilde{B},a)}(\{\alpha_{j,b}\}_{j\in[m],b\in\{0,1\}}) = C(\{\alpha_{j,b} \cdot \tilde{B}_{j,b}\}_{j\in[m],b\in\{0,1\}}) - a$$

When the event E occurs, we claim that the resulting polynomial $q_{(\tilde{B},a)}$ is a non-zero polynomial of degree at most m. This can be easily seen given the decomposition of C in (4). When $q_{(\tilde{B},a)}$ is a non-zero polynomial then by the Schwartz Zippel lemma, its evaluation on uniformly random non-zero inputs $\{\alpha_{j,b}\}_{j\in[m],b\in\{0,1\}}$ is zero with probability at most $m/p-1 \leq 2wm/p$. Therefore, we have

$$Pr[C(\mathsf{Rand}(BP,p)) = a|E] = Pr[q_{\tilde{B}}(\{\alpha_{j,b}\}) = 0|E] \leq \frac{2wm}{p} \tag{6}$$

Combining (6) and (5) and Claim 20, we have $Pr[C(\mathsf{Rand}(BP,p)) = a] \leq 12wm/p$. □

5.5 Achieving Obfuscation for Arbitrary Programs

[GGH$^+$13b] show that any indistinguishability obfuscation scheme for NC^1 can be bootstrapped into an indistinguishability obfuscation scheme for all poly-sized circuits using FHE. That is, they prove the following theorem.

Theorem 21 ([GGH$^+$13b]—informally stated)**.** *Assume the existence of indistinguishability obfuscators $i\mathcal{O}$ for NC^1 and a leveled Fully Homomorphic Encryption scheme with decryption in NC^1. Then there exists an indistinguishability obfuscator $i\mathcal{O}'$ for arbitrary poly-sized circuits.*

Applying their construction to our indisinguishability obfuscator yields an indistinguishability obfuscator for arbitrary polynomial size circuits:

Theorem 22. *Assume the existence of a multilinear encodings that is entropic semantically secure (with respect to simple message samplers) and a leveled Fully Homomorphic Encryption scheme with decryption in NC^1. Then there exists indistinguishability obfuscators for arbitrary poly-sized circuits.*

6 Impossibility of Extractable Semantic Security

In this section we consider a stronger notion of *extractable semantic security* of multilinear encodings, which strengthens the notion of semantic security by requiring that if an attacker A can distinguish between encodings of a constant number of elements, then there exists an *efficient* algebraic strategy that distinguishes the elements; that is, the algebraic operations (i.e., the legal arithmetic circuit queries) needed to distinguish the elements can be efficiently "extracted out". (Recall that, in contrast, in the definition of "plain" semantic security, we allow the algebraic strategy to be computationally unbounded, while restricting it to making only polynomially many queries). We here show that, assuming the existence of a leveled FHE with decryption in NC^1, there do not exist extractable semantically secure multilinear encodings.

Formally, we define a notion of a *computationally respecting* message sampler condition in exactly the same way as the definition of a respecting message sampler (Definition 8), except that we restrict the algebraic attacker to be described by an (oracle) circuit of size at most $p(n)$ (as opposed to being computationally unbounded, but only making $p(n)$ oracle queries). We define *extractable* semantic security in exactly the same way as semantic security except that we only require the message sampler to be computationally respecting message (and define entropic extractable semantic security in the analogous way).
We now have the following theorem.

Theorem 23. *Assume the existence of a leveled Fully Homomorphic Encryption scheme with decryption in NC^1. Then no graded encoding scheme satisfies entropic extractable semantic security.*

Proof. Consider any graded encoding scheme $\mathcal{E}$. To show that $\mathcal{E}$ is not entropic extractable semantically secure we need to show that there exists an entropic computationally respecting message sampler M and PPT adversary A such that A distinguishes between encodings of $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$ where $(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M$.

Our M will sample obfuscations of the following circuit family, that was shown to be unobfuscatable in the virtual black box setting [BGI+01].Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a semantically secure fully homomorphic encryption scheme with ciphertext size $N(\cdot)$; for simplicity of exposition, let us first assume that it is an "unleveled" FHE. For each security parameter n, consider the class of circuits

$$\mathcal{C}_n = \{C_{n,a,b,v,\mathsf{pk},\mathsf{sk},\hat{a}}\}_{a,b\in\{0,1\}^n, v\in\{0,1\}, (\mathsf{pk},\mathsf{sk})\in\mathsf{Gen}(1^n), \hat{a}\in\mathsf{Enc}(pk,a)}$$

taking $N(n)$-bit inputs, where

$$C_{n,a,b,v,\mathsf{pk},\mathsf{sk},\hat{a}}(x) = \begin{cases} (\mathsf{pk}, \hat{a}) & \text{if } x = 0 \\ b & \text{if } x = a \\ v & \text{if } \mathsf{Dec}(\mathsf{sk}, x) = b \\ 0 & \text{otherwise} \end{cases}$$

Then $M(\mathsf{pp})$ operates as follows, given public parameters pp to a graded encoding scheme, containing the security parameter n and the description of the ring $\mathbb{Z}_p$:

- M samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$ and $a, b \leftarrow \{0,1\}^n$ uniformly at random, and computes $\hat{a} = \mathsf{Enc}(\mathsf{pk}, a)$.
- M generates branching programs BP_0 and BP_1 corresponding to $C_{n,a,b,0,\mathsf{pk},\mathsf{sk},\hat{a}}$ and $C_{n,a,b,1,\mathsf{pk},\mathsf{sk},\hat{a}}$ respectively, and computes $\widehat{BP}_0 = \mathsf{Merge}(BP_0, BP_1, 0)$ and $\widehat{BP}_1 = \mathsf{Merge}(BP_0, BP_1, 1)$, each of width 10 and length m. Recall, from Claim 8, that $\widehat{BP}_0$ and $\widehat{BP}_1$ differ only in levels 1 and m, and that $\widehat{BP}_0$ and $\widehat{BP}_1$ are functionally equivalent to BP_0 and BP_1 respectively.
- M samples m random invertible matrices over $\mathbb{Z}_p^{10\times 10}$, $\{R_i\}_{i\in[m]}$ and $2m$ random scalars from $\mathbb{Z}_p$, $\{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}$. M then uses these matrices and scalars to randomize $\widehat{BP}_0$ and $\widehat{BP}_1$ as described by $\mathsf{Rand}(\cdot, p)$ to obtain $\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}$, $\{\alpha_{i,b} \cdot \tilde{B}'_{i,b}\}_{i\in[m],b\in\{0,1\}}$ and $\mathbf{t}$. M further generates the sets $\{S_{i,b}\}_{i\in[m],b\in\{0,1\}}, S_t$ under which to encode these elements by a call to $\mathsf{SetSystem}$.
- M outputs

$$\vec{m}_0 = (\{(\alpha_{1,b} \cdot \tilde{B}_{1,b}, S_{1,b})\}_{b\in\{0,1\}}, \{(\alpha_{m,b} \cdot \tilde{B}_{m,b}, S_{m,b})\}_{b\in\{0,1\}})$$
$$\vec{m}_1 = (\{(\alpha_{1,b} \cdot \tilde{B}'_{1,b}, S_{1,b})\}_{b\in\{0,1\}}, \{(\alpha_{m,b} \cdot \tilde{B}'_{m,b}, S_{m,b})\}_{b\in\{0,1\}})$$
$$\vec{z} = ((\{\alpha_{i,b} \cdot \tilde{B}_{i,b}, S_{i,b})\}_{i\in[m]/\{1,m\},b\in\{0,1\}}, (\mathbf{t}, S_t))$$

  Note that $(\vec{m}_0, \vec{z})$ is identically distributed to $\mathsf{GObf}(\widehat{BP}_0, \mathsf{pp})$ and similarly $(\vec{m}_1, \vec{z})$ is identically distributed to $\mathsf{GObf}(\widehat{BP}_1, \mathsf{pp})$. As a result, by Proposition 2, we have that M is an entropic message sampler.

To show that M is a computationally respecting message sampler, we need to show that no polynomial-size oracle circuit A' can pointwise distinguish the oracles $\mathcal{M}(\vec{m}_0, \vec{z})$ and $\mathcal{M}(\vec{m}_1, \vec{z})$; that is, for every polynomial $p(\cdot)$, there exists a negligible function μ such that for every n, for every pp, for every oracle circuit A' of size at most $p(n)$,

$$\Pr[(\vec{m}_0, \vec{m}_1, \vec{z}) \leftarrow M(\mathsf{pp}) : A'^{\mathcal{M}(\vec{m}_0,\vec{z})} \neq A'^{\mathcal{M}(\vec{m}_1,\vec{z})}] \leq \mu(n)$$

Recall that by Claim 15, the output of $A'^{\mathcal{M}(\vec{m}_0,\vec{z})}$ (resp. $A'^{\mathcal{M}(\vec{m}_1,\vec{z})}$) can be simulated with only oracle access to BP_0 (resp. BP_1), or equivalently, to $C_{n,a,b,0,\mathsf{pk},\mathsf{sk},\hat{a}}$ (resp. $C_{n,a,b,1,\mathsf{pk},\mathsf{sk},\hat{a}}$). In fact, with high probability over the randomness of M, A' and the simulator, the simulator's output is identical to the output of A'. We observe that this simulation can be made *efficient* using the techniques introduced in

[BGK+13] (i.e. by modifying BP_0 and BP_1 to be *dual-input* branching programs and correspondingly changing $\mathsf{SetSystem}$); this requires encodings elements using sets of size 4 (as opposed to 2 as in our original construction). Let this efficient simulator be Sim.

We would now like to argue that with high probability over the randomness of M and Sim, $\mathsf{Sim}^{BP_0} = \mathsf{Sim}^{BP_1}$. Recall that the circuits $C_{n,a,b,0,\mathsf{pk},\mathsf{sk},\hat{a}}$ (equivalent to BP_0) and $C_{n,a,b,1,\mathsf{pk},\mathsf{sk},\hat{a}}$ (equivalent to BP_1) differ only on inputs x for which $\mathsf{Dec}(\mathsf{sk}, x) = b$ (on these inputs $C_{n,a,b,0,\mathsf{pk},\mathsf{sk},\hat{a}}(x) = 0$, whereas $C_{n,a,b,1,\mathsf{pk},\mathsf{sk},\hat{a}}(x) = 1$). Since b was randomly chosen from an exponentially large set of values, to find such an input with noticeable probability, Sim must query one of the circuits on input a with noticeable probability, otherwise its view is independent of b. However, if the original ciphertext $\hat{a}$ is an encryption of 0 instead of a, then the view of Sim is independent of a, and thus Sim can only query a with negligible probability. Thus by the semantic security of the FHE scheme, the probability that Sim can query a when given BP_0 or BP_1 is negligible. This implies that the outputs of Sim^{BP_0} and Sim^{BP_1} differ with only negligible probability.

We now have that :

- $A'^{\mathcal{M}(\vec{m}_0,\vec{z})} = \mathsf{Sim}^{BP_0}$, except with negligible probability;
- $\mathsf{Sim}^{BP_0} = \mathsf{Sim}^{BP_1}$, except with negligible probability;
- $\mathsf{Sim}^{BP_1} = A'^{\mathcal{M}(\vec{m}_1,\vec{z})}$, except with negligible probability.

By a union bound, we have that $A'^{\mathcal{M}(\vec{m}_0,\vec{z})} = A'^{\mathcal{M}(\vec{m}_1,\vec{z})}$, except with negligible probability. Thus M must be a computationally respecting sampler. Finally, it follows using identically the same arguement as in Section 5.4 that the message sampler satisfies the required high-entropy condition and thus is an entropic computationally respecting message sampler.

Now we will show an *nuPPT* adversary A that distinguishes between encodings of $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$ when encoded under sets $(\vec{S}, \vec{T}) \leftarrow \mathsf{SetSystem}(m, N, \mathsf{inp})$, where inp is the labelling function for the branching programs BP_0 and BP_1. Note that given encodings of one of $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$, A in fact receives either $\mathsf{Obf}(\widehat{BP}_0)$ or $\mathsf{Obf}(\widehat{BP}_1)$. Let us refer to this input to A as O.

A evaluates O on input 0 to receive $(\mathsf{pk}, \hat{a})$, and then simply homomorphically evaluates O on the ciphertext $\hat{a}$ in order to generate a valid encryption of the hidden value b, and then feeds this new ciphertext back into O to reveal the secret bit v, and then outputs v. Thus A succeeds in distinguishing $(\vec{m}_0, \vec{z})$ and $(\vec{m}_1, \vec{z})$ with probability 1. Additionally, note that since O is a constant-width branching program, O can be computed by a NC^1 circuit, thus for this argument it suffices to use a leveled FHE.

We thus have that no graded encoding scheme can satisfy extractable semantic security. $\square$

Note that the above proof in fact rules out also entropic extractable semantical security with respect to a slight relaxation of simple message samplers, where instead of restricting to encodings under sets of size 2, we restrict to sets of size 4.

Let us end this section by remarking that we do not view Theorem 3 as an indication of the impossibility of "plain" semantical security for multilinear encodings as the notions of extractable and "plain" semantical security are *qualitatively* very different. (Indeed, extractability assumptions have recently been shown to be problematic is various different contexts [HT98, BCCT12, BCPR13, BP13, GGHW13], but as far as we now this is the first impossibility result for an extractability assumption based on standard hardness assumptions.)

7 Acknowledgments

We are very grateful to Omer Paneth, Ran Canetti, Kai-Min Chung, Sanjam Garg, Craig Gentry, Shai Halevi, Abhi Shelat and Hoeteck Wee for many helpful comments. We are especially gratefeul to Shai for pointing out the connection between semantical security for multilinear encodings and the "uber" assumption for bilinear maps of [BBG05], and for several very useful conversations about multilinear encodings and the security of the [GGH13a] constructions. Thanks a lot!

References

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. 2013.

[Bar86] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1. In *STOC*, pages 1–5, 1986.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer Berlin Heidelberg, 2005.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

[BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. Indistinguishability obfuscation vs. auxiliary-input extractable functions: One must fall. Technical report, Cryptology ePrint Archive, Report 2013/641, 2013.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology CRYPTO 2001*, pages 1–18. Springer, 2001.

[BGK$^+$13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. Cryptology ePrint Archive, Report 2013/631, 2013.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. 2013.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *TCC*, pages 1–25, 2014.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011.

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Technical report, Cryptology ePrint Archive, Report 2013/642, 2013. http://eprint. iacr. org, 2013.

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

[CLT^{+}13] Jean-Sébastien Coron, Tancrède Lepoint, Mehdi Tibouchi, et al. Practical multilinear maps over the integers. In *CRYPTO 2013-33rd Annual Cryptology Conference Advances in Cryptology*, volume 8042, pages 476–493, 2013.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pages 416–426, 1990.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, 2009.

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology–EUROCRYPT 2013*, pages 1–17. Springer, 2013.

[GGH^{+}13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *Proc. of FOCS 2013*, 2013.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014.

[GGHW13] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. Technical report, Cryptology ePrint Archive, Report 2013/860, 2013. 6, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the 45th Annual ACM Symposium on Symposium on Theory of Computing*, STOC '13, pages 467–476, New York, NY, USA, 2013. ACM.

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

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *Advances in Cryptology–ASIACRYPT 2000*, pages 443–457. Springer, 2000.

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Technical report, Cryptology ePrint Archive, Report 2013/509, 2013. http://eprint. iacr. org, 2013.

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In *Advances in Cryptology CRYPTO'98*, pages 408–423. Springer, 1998.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 20–31. ACM, 1988.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437. ACM, 1990.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.

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

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. Technical report, IACR Cryptology ePrint Archive, 2013: 454, 2013.

A Technical Lemma

Claim 24. *Fix $m, w \in \mathbb{N}$, and let $p \in \mathbb{N}$ be a prime. Let $\mathcal{D}_0$ be the following distribution:*

$$\mathcal{D}_0 = \{\{R_i\}_{i\in[m]}, \{\alpha_{i,b}\}_{i\in[m],b\in\{0,1\}}\}$$

where each R_i is a uniformly random invertible matrix in $\mathbb{Z}_p^{w\times w}$ (i.e $det(R_i) \neq 0$, and each $\alpha_{i,b}$ is a uniformly random non-zero scalar in $\mathbb{Z}_p$.

Let $\mathcal{D}_1$ be a distribution defined identically to $\mathcal{D}_0$, except with each R_i being a uniformly random (not necessarily invertible) matrix in $\mathbb{Z}_p^{w\times w}$, and each $\alpha_{i,b}$ a uniformly random (not necessarily non-zero) scalar in $\mathbb{Z}_p$.
Then:

$$\Delta(\mathcal{D}_0, \mathcal{D}_1) \leq 8wm/p$$

where $\Delta(\mathcal{D}_0, \mathcal{D}_1)$ denotes the statistical distance between distributions $\mathcal{D}_0$ and $\mathcal{D}_1$.

Proof. Note that $\mathcal{D}_0$ and $\mathcal{D}_1$ are each uniformly distributed on their respective supports, and that $\mathsf{supp}(\mathcal{D}_0) \subseteq \mathsf{supp}(\mathcal{D}_1)$. Then the statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ can be computed as follows:

$$\begin{aligned}
\Delta(\mathcal{D}_0, \mathcal{D}_1) &= \sum_{d\in\mathsf{supp}(\mathcal{D}_0)\cup\mathsf{supp}(D_1)} |\Pr[\mathcal{D}_0 = d] - \Pr[\mathcal{D}_1 = d]| \\
&= \sum_{d\in\mathsf{supp}(\mathcal{D}_0)} |\Pr[\mathcal{D}_0 = d] - \Pr[\mathcal{D}_1 = d]| + \sum_{d\in\mathsf{supp}(\mathcal{D}_1)\setminus\mathsf{supp}(\mathcal{D}_0)} |\Pr[\mathcal{D}_1 = d]| \\
&= \sum_{d\in\mathsf{supp}(\mathcal{D}_0)} |\frac{1}{|\mathsf{supp}(\mathcal{D}_0)|} - \frac{1}{|\mathsf{supp}(\mathcal{D}_1)|}| + \sum_{d\in\mathsf{supp}(\mathcal{D}_1)\setminus\mathsf{supp}(\mathcal{D}_0)} |\frac{1}{|\mathsf{supp}(\mathcal{D}_1)|}| \\
&= (|\mathsf{supp}(\mathcal{D}_0)| \cdot |\frac{1}{|\mathsf{supp}(\mathcal{D}_0)|} - \frac{1}{|\mathsf{supp}(\mathcal{D}_1)|}|) + (|\mathsf{supp}(\mathcal{D}_1) \setminus \mathsf{supp}(\mathcal{D}_0)| \cdot |\frac{1}{|\mathsf{supp}(\mathcal{D}_1)|}|) \\
&= 2 \cdot (1 - \frac{|\mathsf{supp}(\mathcal{D}_0)|}{|\mathsf{supp}(\mathcal{D}_1)|})
\end{aligned}$$

But notice that $(1 - \frac{|\mathsf{supp}(\mathcal{D}_0)|}{|\mathsf{supp}(\mathcal{D}_1)|})$ can be interpreted as $\Pr[\exists i \in [m], b \in \{0,1\} : \det(R_i) = 0 \vee \alpha_{i,b} = 0]$. For each $i \in [m]$, the probability $det(R_i) = 0$ can be bounded by applying the Schwartz-Zippel lemma to the $det(\cdot)$, which is a polynomial of degree w. Thus we have that $\Pr[det(R_i) = 0] \leq w/p$. Further, each $\alpha_{i,b}$ is zero with probability $1/p$. Hence, applying a union bound, we have that

$$\begin{aligned}
\Delta(\mathcal{D}_0, \mathcal{D}_1) &= 2 \cdot (1 - \frac{|\mathsf{supp}(\mathcal{D}_0)|}{|\mathsf{supp}(\mathcal{D}_1)|}) \\
&\leq 2 \cdot (2m/p + mw/p) \\
&\leq 8wm/p
\end{aligned}$$

□

B Proof of Lemma 17

In this section, we prove Lemma 17, restated below for clarity:

Lemma 17. *Fix $m, n, w \in \mathbb{N}$ and $\mathsf{inp} : [m] \to [n]$. Let $\vec{S} = \mathsf{SetSystem}(m, n, \mathsf{inp}) = (\{S_{i,b}\}_{i\in[m],b\in\{0,1\}}, S_t)$, and let C be any weakly $\vec{S}$-respecting arithmetic circuit whose output wire is tagged with $T \subseteq [k]$. Then there exists a set $U \subseteq \{0, 1, *\}^m$ such that for every branching program BP of width w and length m on n input bits, with input tagging function inp, every prime p, and every $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in m,b\in\{0,1\}}, \mathbf{t}) \leftarrow \mathsf{Rand}(BP, p)$,*

(i)

$$C(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) \equiv \sum_{u\in U} C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t})$$

where each C_u is a weakly $\vec{S}$-respecting arithmetic circuit, whose input wires are tagged only with sets $\in \{S_{i,u[i]}\}_{i\in[m]:u[i]\neq*} \cup \{S_t\}$, and whose output wire is tagged with T.

(ii) *Each C_u above is the sum of several "monomial" circuits, where each monomial circuit performs only multiplications of elements in $(\{\alpha_{i,b} \cdot \tilde{B}_{i,b}\}_{i\in m,b\in\{0,1\}}, \mathbf{t})$, is weakly $\vec{S}$-respecting, and has output wire tagged with T.*

(iii) *For each C_u above,*

$$C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i\in[m]:u[i]\neq*}, \mathbf{t})$$

where p_u is some polynomial, and $\alpha_u = (\prod_{i\in[m]:u[i]\neq*} \alpha_{i,u[i]})$. Furthermore, when p_u is viewed as a sum of monomials, each monomial contains exactly one entry from each $\tilde{B}_{i,u[i]}$ such that $u[i] \neq *$, and possibly one entry from $\mathbf{t}$. Further, p_u can be computed by a weakly $\vec{S}$-respecting circuit whose output wire is tagged with T.

Proof. **Part (i)** We begin by expressing the circuit C as a polynomial in variables $(\{\alpha_{i,b}\cdot\tilde{B}_{i,b}\}_{i\in m,b\in\{0,1\}}, \mathbf{t})$, in the form of a sum of monomials (possibly exponentially many). We do so recursively: we associate each wire w of the circuit with a multiset S_w of pairs of monomials and signs ("+1" or "-1"), such that the sum of the monomials multiplied by their respective signs computes the same value as the value computed by the circuit at that wire. We eventually output the multiset of monomial pairs corresponding to the output wire. We compute the sets of monomials as follows:

- Any input wire of the circuit reading input variable v can be represented as the set $\{(v, +)\}$.
- The output wire of an addition gate can be represented as the union of the multisets of monomial pairs representing the gates left and right children.
- The output wire of an subtraction gate can be similarly represented as the union of the multisets of the gate's left input wire, and of its right input wire with the "sign" component of every pair negated (from "+1" to "-1" and vice versa), to reflect subtraction.
- For the output wire of a multiplication gate, for each pair (M_1, s_1) in the multiset of its left input and each pair (M_2, s_2) in the multiset of its right input, we add $(M_1 \cdot M_2, s_1 \cdot s_2)$ to the multiset of the output wire.

We note that it holds inductively in the above process that the sum of the monomials in the multiset associated with each wire w in C, multiplied by its appropriate sign, equals the value computed on that wire w.

We also show that each monomial in the set corresponding to a wire can be computed by a weakly $\vec{S}$-respecting circuit whose output wire has the same tag as the wire. This can again be seen inductively:

- This property holds at any input wire of C, since the only monomial in the set can be computed using the input wire itself as the “monomial circuit".
- This property also holds at any output wire of an addition or subtraction gate, since the circuit corresponding to any monomial in this wire's set is the same as the circuit for the monomial from the corresponding incoming wire to the gate.
- Finally, at the output wire of a multiplication gate G, for any monomial M in this wire's set computed as the product of monomials $M1$ and $M2$, the circuit for M is simply the circuit for each of $M1$ and $M2$, joined by a multiplication gate. Since G performs a set respecting multiplication, and the output wires of $M1$ and $M2$'s circuits have the same tags as the input wires of G, we have that the multiplication joining $M1$ and $M2$'s circuits to produce M's circuit is set-respecting, and so the circuit corresponding to M is a weakly $\vec{S}$-respecting circuit whose output wire has the same tag as the output wire of G.

Thus each of the monomials in the decomposition of C can be represented as a weakly set-respecting arithmetic circuit with output wire tagged with T, where this circuit simply multiplies together all terms in the monomial in some order, and performs no additions. Finally, the tags of the input wires of these monomial circuits must be mutually disjoint, otherwise the monomial circuit would perform a non-set-respecting multiplication at some level.

We label each monomial M with an element $u \in \{0, 1, *\}^m$, where $u[i] = b$ if $S_{i,b}$ is the label on one of input wires in M's circuit representation, and $u[i] = *$ if neither $S_{i,0}$ and $S_{i,1}$ are labels on any of M's input wires. We note that no monomial can have both $S_{i,0}$ and $S_{i,1}$ on its input wires because these two sets are not disjoint, and the tags of the input wires of the monomial circuits must be mutually disjoint.

We now let C_u be the circuit representing the subtraction of all momonials in the the decomposition of C labelled with u and sign (-1) from the sum of all momonials in the the decomposition of C labelled with u and sign $(+1)$. Since each monomial can be represented as a weakly set-respecting circuit with output wire tagged with T, adding several monomials together is a set-respecting operation, as is subtracting several monomials from the sum, and thus each C_u is a weakly set-respecting circuit. Further, since each monomial circuit has output wire tagged with T, each C_u also has output wire tagged with T. Further, by the way we labelled each monomial, each of the input wires of C_u is tagged only with sets $\in \{S_{i,u[i]}\}_{i\in[m]:u[i]\neq *} \cup \{S_t\}$. Finally, if we sum over all the u, we capture all the monomials in the decomposition of C multiplied by their respective signs, so we have that $\sum_u C_u = C$.

Part (ii) We observe that by construction of C_u, it is a sum of several monomial circuits each of which performs only multiplications of its inputs, is weakly $\vec{S}$-respecting, and has output wire tagged with T.

Part (iii) From part (ii), we have that for each C_u, it is a sum of several monomial circuits each of which performs only multiplications of its inputs, is weakly $\vec{S}$-respecting, and has output wire tagged with T. Furthermore, for each such monomial circuit the input tags are drawn from sets $\in \{S_{i,u[i]}\}_{i\in[m]:u[i]\neq *} \cup \{S_t\}$. In fact, each of these monomials must contain exactly one input wire tagged with each of the sets in $\{S_{i,u[i]}\}_{i\in[m]:u[i]\neq *}$, and exactly one set tagged with S_t if and only if $S_t \subseteq T$. This means that each of these monomials is the product of one element chosen from each of the matrices $(\{\alpha_{i,u[i]} \cdot \tilde{B}_{i,u[i]}\}_{i\in m:u[i]\neq *}$, and possibly one element from $\mathbf{t}$. Thus each monomial in the decomposition of C_u has a common factor of $\alpha_u = (\prod_{i\in[m]:u[i]\neq *} \alpha_{i,u[i]})$.

We can now write C_u as a polynomial (namely the sum of its monomials multiplied by their respective signs), and by factoring α_u from each of it monomials and letting p_u be the remaining polynomial, we have, as required, that

$$C_u(\{\alpha_{i,b}\tilde{B}_{i,b}\}_{i\in[m],b\in\{0,1\}}, \mathbf{t}) = \alpha_u \cdot p_u(\{\tilde{B}_{i,u[i]}\}_{i\in[m]:u[i]\neq *}, \mathbf{t})$$

Finally, we note that computing p_u is the same as computing C_u if the alphas are set to 1. Since C_u is $\vec{S}$-respecting, we thus have that p_u can be computed by a weakly $\vec{S}$-respecting circuit whose output wire is tagged with T. $\square$