On Removing Graded Encodings from Functional Encryption*

Nir Bitansky† Huijia Lin‡ Omer Paneth§

April 7, 2017

Abstract

Functional encryption (FE) has emerged as an outstanding concept. By now, we know that beyond the immediate application to computation over encrypted data, variants with *succinct ciphertexts* are so powerful that they yield the full might of indistinguishability obfuscation (IO). Understanding how, and under which assumptions, such succinct schemes can be constructed has become a grand challenge of current research in cryptography. Whereas the first schemes were based themselves on IO, recent progress has produced constructions based on *constant-degree graded encodings*. Still, our comprehension of such graded encodings remains limited, as the instantiations given so far have exhibited different vulnerabilities.

Our main result is that, assuming LWE, *black-box constructions* of *sufficiently succinct* FE schemes from constant-degree graded encodings can be transformed to rely on a much better-understood object — *bilinear groups*. In particular, under an *über assumption* on bilinear groups, such constructions imply IO in the plain model. The result demonstrates that the exact level of ciphertext succinctness of FE schemes is of major importance. In particular, we draw a fine line between known FE constructions from constant-degree graded encodings, which just fall short of the required succinctness, and the holy grail of basing IO on better-understood assumptions.

In the heart of our result, are new techniques for removing ideal graded encoding oracles from FE constructions. Complementing the result, for weaker ideal models, namely the generic-group model and the random-oracle model, we show a transformation from *collusion-resistant* FE in either of the two models directly to FE (and IO) in the plain model, without assuming bilinear groups.

*An extended abstract of this paper appears in the proceedings of EUROCRYPT 2017.

†MIT, nirbitan@csail.mit.edu. Supported by NSF Grants CNS-1350619 and CNS-1414119, and the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226. Part of this research was done while visiting Tel Aviv University and supported by the Leona M. & Harry B. Helmsley Charitable Trust and Check Point Institute for Information Security.

‡UCSB, rachel.lin@cs.ucsb.edu. partially supported by NSF grants CNS-1528178 and CNS-1514526.

§MIT, omerpa@gmail.com.

Contents

1 Introduction

Functional Encryption (FE) is a fascinating object. It enables fine-grained control of encrypted data, by allowing users to learn only specific functions of the data. This ability is captured trough the notion of *function keys*. A function key SK_f, associated with a function f, allows to partially decrypt a ciphertext CT_x encrypting an input x in a way that reveals $f(x)$ and nothing else.

A salient aspect of FE schemes is their *ciphertext succinctness*. Focusing on the setting of (indistinguishability-based) *single-key* FE where only one function key SK_f is supported, we say that an FE scheme is *weakly succinct* if the ciphertext size scales *sub-linearly* in the size of the circuit f; namely,

$$|\mathsf{CT}_x| \leq |f|^\gamma \cdot \mathrm{poly}(|x|), \qquad \text{for } \textit{some} \text{ constant } \textit{compression factor } \gamma < 1.$$[1]

While non-succinct single-key FE schemes (where we allow the size of ciphertexts to grow polynomially with $|f|$) are equivalent to public-key encryption (or just one-way functions, in the secret-key setting) [SS10, GVW12], weakly succinct schemes are already known to be extremely strong. In particular, subexponentially-secure weakly-succinct FE for functions in $\mathbf{NC}^1$ implies indistinguishability obfuscation (IO) [BGI+12, AJ15, BV15], and has far reaching implications in cryptography and beyond (*e.g.*, [GGH+13c, GGHR14, SW14, BPR15, BGJ+16, BZ16]).[2]

Thus, understanding how, and under which assumptions, weakly-succinct FE can be constructed has become a central question in cryptographic research. While schemes for Boolean functions in $\mathbf{NC}^1$ have been constructed from LWE [GKP+13], the existence of such FE scheme for non-Boolean functions (which is required for the above strong implications) is still not well-founded, and has been the subject of a substantial body of work. The first construction of general purpose FE that achieves the required succinctness relied itself on IO [GGH+13c]. Subsequent constructions were based on the algebraic framework of *multilinear graded encodings* [GGH13a]. Roughly speaking, this framework extends the traditional concept of *encoding in the exponent* in groups. It allows *encoding* values in a field (or ring), evaluating polynomials of a certain bounded *degree* d over the encoded values, and testing whether the result is zero.

Based on graded encodings of polynomial degree Garg, Gentry, Halevi, and Zhandry [GGHZ16] constructed *unbounded-collusion* FE, which in turn is known to lead to weakly succinct FE [AJS15, BV15]. Starting from the work of Lin [Lin16a], several works [LV16, AS16, Lin16b] have shown that assuming also pseudorandom generators with constant locality, weakly-succinct FE can be constructed based on *constant-degree* graded encodings under simple assumptions like asymmetric DDH. However, these constructions require constant degree $d \geq 5$.

Despite extensive efforts, our understanding of graded encodings of any degree larger than two is quite limited. Known instantiations are all based on little-understood lattice problems, and have exhibited different vulnerabilities [GGH13a, CHL+15, CGH+15, BWZ14, MSZ16]. In contrast, bilinear group encodings [BF01, Jou02], akin to degree-2 graded encodings, have essentially different instantiations based on elliptic curve groups, which are by now quite well understood and considered standard. Bridging the gap between degree 2 and degree $d > 2$ is a great challenge.

Our Main Result in a Nutshell: Size Matters. We show that the exact level of succinctness in FE schemes has a major impact on the latter challenge. Roughly speaking, we prove that *black-box constructions* [RTV04] of weakly-succinct FE from degree-d graded encodings, with compression factor $\gamma < \frac{1}{d}$, can be transformed

[1] Here *weak* succinctness is in contrast to *full* succintness, where the ciphertext size does not depend at all on the function size.

[2] Formally, [AJ15, BV15] require that not only the ciphertext is succinct, but also the encryption circuit itself. This difference can be bridged assuming LWE [LPST16b], and for simplicity is ignored in this introduction. Our results will anyhow rely on LWE.

to rely *only on bilinear groups*. Specifically, assuming LWE [3] and for any constant ε, starting from $\frac{1}{d+\varepsilon}$-succinct FE in the *ideal degree-d graded encoding model*, we construct weakly-succinct FE in the *ideal bilinear model*.

The ideal graded encoding model generalizes the classical generic-group model [Sho97]. In this model, the construction as well as the adversary perform all graded encoding operations through an *ideal oracle*, without access to an explicit representation of encoded elements. Having this ideal model as a starting point allows capturing a large class of constructions and assumptions, as it models *perfectly secure* graded encodings. Indeed, the FE schemes in [Lin16a, LV16, AS16, Lin16b] can be constructed and proven secure in this model.

The resulting construction from ideal bilinear encodings can further be instantiated in the plain model using existing bilinear groups, and proven secure under an *über assumption* on bilinear groups [BBG05, Boy08]. In particular, assuming also subexponential-security, it implies IO in the plain model.

How Close Are We to IO from Bilinear Maps? Existing weakly-succinct FE schemes in the ideal constant-degree model [Lin16a, LV16, AS16, Lin16b] have a compression factor $\gamma = C/d$, for some absolute constant $C > 1$. Thus, our result draws a fine line that separates known FE constructions based on constant-degree graded encodings and constructions that would already take us to the promised land of IO based on much better-understood mathematical objects. Crossing this line may very well require a new set of techniques. Indeed, one may also interpret our result as a negative one, which puts a barrier on black-box constructions of FE from graded encodings.

Discussion: Black-Box vs Non-Black-Box Constructions. For IO schemes (rather than FE), a combination of recent works [PS16, BV16] demonstrates that black-box constructions from constant-degree graded encodings are already very powerful. They show that any IO construction relative to a constant-degree oracle can be converted to a construction in the plain model (under standard assumptions, like DDH). Since weakly-succinct FE schemes imply IO, we may be lead to think that weakly-succinct black-box constructions of FE from constant-degree graded encodings would already imply IO in the plain model from standard assumptions. Interestingly, this is not the case.

The crucial point is that the known transformations from FE to IO [AJ15, BV15] are *non-black-box*, they use the code of the underlying FE scheme, and thus do not *relativize* with respect to graded encoding oracle. That is, we do not know how to move from an FE scheme based on graded encodings to an IO scheme that uses graded encodings in a black-box way. Indeed, if there existed such a black-box transformation between FE and IO, then combining [PS16, MMN16, BV16, Lin16a, LV16], IO in the plain model could be constructed from standard assumptions.

Instead, we show how to remove constant-degree oracles directly from FE. Our transformation relies on new techniques that are rather different than those previously used for removing such oracles from IO.

1.1 Our Results in More Detail

We now describe our results in further detail. We start by describing the ideal graded encoding model and the ideal bilinear encoding model more precisely.

The Ideal Graded Encoding Model. A graded encoding [GGH13b] is an encoding scheme for elements of some field.[4] The encoding supports a restricted set of homomorphic operations that allow one to evaluate

[3]More precisely, we need to assume the hardness of LWE with subexponential modulus-to-noise ratio. For simplicity, we ignore the parameters of LWE in this introduction; see Section 7.3 for more details.

[4]For ease of exposition, we consider graded encodings over fields. Our results can also be obtained with any commutative ring in which it is computationally hard to find non-unit elements.

certain polynomials over the encoded field elements and test whether these polynomials evaluate to zero or not. Every field element is encoded with respect to a *label* (sometimes called the *level* of the encoding). For a given sequence of encodings, their labels control which polynomials are valid and can be evaluated over the encodings. The *degree* of the graded encoding is the maximal degree of any valid polynomial.

In the ideal graded encoding model, explicit encodings are replaced by access to an oracle that records the encoded field elements and provides an interface to perform operations over the elements. Different formalizations of such ideal graded encoding oracles exist in the literature (*e.g.* [BR14, BGK+14, AB15, PS16]) and differ in details. In this work, we follow the model of Pass and shelat in [PS16].

The ideal graded encoding oracle $\mathcal{M}$ is specified by a field $\mathbb{F}$ and a validity predicate V operating on a polynomial and labels taken from a set $\mathbb{L}$. The oracle $\mathcal{M} = (\mathbb{F}, V)$ provides two functions — encoding and zero-testing.

Encoding: Given a field element $\xi \in \mathbb{F}$ and a label $\ell \in \mathbb{L}$ the oracle $\mathcal{M}$ samples a sufficiently long random string r to create a *handle* $h = (r, \ell)$. It records the pair (h, ξ) associating the handle with the encoded field element.

Zero-testing: a query to $\mathcal{M}$ consists of a polynomial p and a sequence of handles $h_1, \cdots, h_m$ where h_i encodes the field elements ξ_i relative to label ℓ_i. $\mathcal{M}$ tests if the polynomial and the labels satisfy the validity predicate and whether the polynomial vanishes on the corresponding field elements. That is, $\mathcal{M}$ returns `true` if and only if $V(p, \ell_1, \cdots \ell_m) = \texttt{true}$ and $p(\xi_1, \cdots \xi_m) = 0$.

Like in [PS16], we restrict attention to well-formed validity predicates. For such predicates, a polynomial p is valid with respect to labels $\ell_1, \cdots, \ell_m$, if and only if every monomial Φ in p is valid with respect to the labels of the handles that Φ acts on. Indeed, existing graded encodings all consider validity predicates that are *well-formed*.[5]

The Ideal Bilinear Encoding Model. The ideal bilinear encoding model corresponds to the ideal graded encoding model where valid polynomials are of degree at most two. We note that in the ideal graded encoding model described above, encoding is a randomized operation. In particular, encoding the same element and label (ξ, ℓ) twice gives back two different handles. In contrast, traditional instantiations of the ideal bilinear encoding model are based on bilinear pairing groups (such as elliptic curve groups) where the encoding is a deterministic function. We can naturally capture such instantiations, by augmenting the ideal bilinear encoding model to use a *unique handle* for every pair of field element and label (as done for instance in [AB15, Zim15, LV16]).

The Main Result. Our main result concerns FE schemes in the ideal graded encoding model. In such FE schemes, all algorithms (setup, key derivation, encryption, and decryption), as well as all adversaries against the scheme, have access to a graded encoding oracle $\mathcal{M}$. We show:

Theorem 1.1 (Informal)**.** *Assume the hardness of LWE. For any constants $d \in \mathbb{N}$ and $\gamma \leq \frac{1}{d}$, any γ-succinct secret-key FE scheme for **P/poly**, in the ideal degree-d graded encoding model, can be transformed into a weakly-succinct public-key FE scheme for **P/poly** in the ideal bilinear encoding model.*

IO in the Plain Model under an Über Assumption. Our main transformation results in a weakly-succinct public-key FE scheme in the ideal bilinear encoding model. By instantiating the ideal bilinear encoding oracle

[5]In the body, we make another structural requirement on validity predicates called *decomposability*. This requirement is somewhat more technical, but is also satisfied by all known formulations of graded encodings. For the simplified technical exposition in this introduction it can be ignored. See further details in Definition 3.3.

with concrete bilinear pairing groups, we get a corresponding FE scheme in the plain model. For security to hold, we make an über assumption [BBG05] on the bilinear groups. An über assumption essentially says that two encoded sequences of elements in the plain model can be distinguished only if they are also distinguishable in the ideal model. There are no known attacks on the über security of existing instantiations of bilinear pairing groups.

Since weakly-succinct public-key FE with subexponential security in the plain model implies IO we deduce the following corollary

Corollary 1.1 (Informal)**.** *Assume subexponential hardness of LWE and bilinear groups with subexponential über security. For any constants $d \in \mathbb{N}, \gamma < \frac{1}{d}$, any subexponentially-secure, γ-succinct, secret-key FE for **P/poly** in the ideal degree-d graded encoding model, can be transformed into an IO scheme for **P/poly** in the plain model.*

FE in Weaker Ideal Models. We also consider FE schemes in ideal models that are weaker than the ideal bilinear encoding model. Specifically, we consider the generic-group model (that corresponds to the ideal degree-1 graded encoding model) and the random-oracle model. We give transformations from FE in these models directly to FE in the plain model *without relying on bilinear encodings*.

In the transformation given by Theorem 1.1, from the ideal constant-degree graded encoding model to the ideal bilinear encoding model, we considered the notion of single-key weakly succinct FE. In contrast, our transformations from the generic-group model and the random-oracle model to the plain model require that we start with a stronger notion of *collusion-resistant FE*. Collusion-resistance requires security in the presence of an *unbounded* number of functional keys. Crucially, ciphertexts are required not to grow with the number of keys (but are allowed to grow polynomially in the size of the evaluated functions).

Collusion-resistant FE is known to imply weakly-succinct FE through a black-box transformation [AJS15, BV15]. In the converse direction, only a non-black-box transformation is known [GS16, LM16], and therefore we cannot apply it to ideal model constructions of FE.

Theorem 1.2 (Informal)**.** *Assume the hardness of LWE. Any collusion-resistant secret-key FE scheme in the generic-group model, or in the random-oracle model, can be transformed into a collusion-resistant public-key FE scheme in the plain model.*

1.2 Our Techniques

We next give an overview of the main ideas behind our degree-reduction transformation given by Theorem 1.1.

Can We Adopt Techniques from IO? As already mentioned, we do not know how to transform FE schemes into IO schemes in a black-box way. Thus, we cannot rely directly on existing results that remove ideal oracles from IO [PS16]. Furthermore, trying to import ideas from these results in the IO setting to the setting of FE encounters some inherent difficulties, which we now explain.

Roughly speaking, removing ideal oracles from IO is done as follows. Starting with a scheme in an ideal oracle model, we let the obfuscator emulate the oracle by itself and publish, together with the obfuscated circuit, some *partial view* of the self-emulated oracle. This partial view is on one hand, sufficient to preserve the functionality of the obfuscated circuit on most inputs, and on the other hand, does not compromise security. The partial view is obtained by evaluating the obfuscation on many random inputs (consistently with the self-emulated oracle), observing how evaluation interacts with the oracle, and performing a certain *learning process*. Arguing that the published partial view does not compromise security crucially relies on the fact that *evaluating the obfuscated program is a public procedure that does not share any secret state with the obfuscator*.

The setting of FE, however, is somewhat more complicated. Here rather than an evaluator we have a decryptor that given a function key SK_f and ciphertext CT encrypting x, should be able to compute $f(x)$. In contrast to the evaluator in obfuscation, the state of the decryptor is *not publicly samplable*. Indeed, generating function keys SK_f for different functions requires knowing a master secret key. Accordingly, it is not clear how to follow the same approach as before.

XIO instead of IO. Nevertheless, we observe that there is a way to reduce the problem to a setting much more similar to IO. Specifically, *there exists [BNPW16] a black-box transformation from FE to a weaker version of IO called XIO*. XIO [LPST16a], which stands for *exponentially-efficient IO*, allows the obfuscation and evaluation algorithms to run in exponential time $2^{O(n)}$ in the input size n, and only requires that the *size* of an obfuscation $\widetilde{C}$ of a circuit C is weakly subexponenetial in n:

$$|\widetilde{C}| \leq 2^{\gamma n} \cdot \mathrm{poly}(|C|) \qquad \text{for some constant } \textit{compression factor } \gamma < 1 \ .$$

Despite this inherent inefficiency, [LPST16a] show that XIO for *logarithmic-size* inputs implies IO assuming subexponential hardness of LWE. A natural direction is thus to try and apply the techniques used to remove oracles from IO to remove the same oracles also from XIO; indeed, if this can be done, such oracles can also be removed from FE, due to the black-box transformation between the two.

This, again, does not work as is. The issue is that the transformations removing degree-d graded encoding oracle from IO may blow up the size of the original obfuscation from $|\widetilde{C}|$ in the oracle model to roughly $|\widetilde{C}|^{2d}$ in the plain model. However, the known black-box construction of XIO from FE [BNPW16] is not sufficiently compressing to account for this blowup. Even starting from FE with great compression, say $\gamma_{\mathsf{FE}} < d^{-10}$, the resulting XIO has a much worst compression factor $\gamma_{\mathsf{XIO}} > 1/2$. In particular, composing the two would result in a useless plain model obfuscation of exponential size $2^{n \cdot d}$.

Motivating our Solution. To understand our solution, let us first describe an over-simplified candidate transformation for reducing XIO with constant-degree graded encoding oracles to XIO with degree-1 oracles (akin to the generic-group model). This transformation will suffer from the same size blowup of the transformations mentioned above.

For simplicity of exposition, we first restrict attention to XIO schemes with the following simple structure (as we shall see later, general XIO schemes may not have this structure):

- Any obfuscated circuit $\widetilde{C}$ consists of a set of handles $h_1, \dots, h_m$ corresponding to field elements $\xi_1, \dots, \xi_m$ encoded during obfuscation, under certain labels $\ell_1, \dots, \ell_m$.
- Evaluation on any given input x consists of performing valid zero-tests over the above handles, which are given by degree-d polynomials $p_1, \dots, p_k$.

A simple idea to reduce the degree-d oracle to a linear oracle is to change the obfuscation algorithm so that it computes ahead of time the field elements ξ_Φ corresponding to all valid degree-d monomials $\Phi(\xi_1, \dots, \xi_m) = \prod_{i \in [d]} \xi_{j_i}$. Then, rather than using the degree-d oracle, it uses the linear oracle to encode the field elements ξ_Φ, and publishes the corresponding handles $\{h_\Phi\}_\Phi$. Evaluation is done in a straight forward manner by writing any zero-test polynomial p of degree d as a linear function in the corresponding monomials

$$p(\xi_1, \cdots, \xi_m) = \sum_{\Phi} \alpha_\Phi \Phi(\xi_1, \dots, \xi_m) \ ,$$

and making the corresponding zero-test query $L_p(\{h_\Phi\}) := \sum_\Phi \alpha_\Phi h_\Phi$ to the linear oracle.

Indeed, the transformation blows up the size of the obfuscated circuit from roughly m, the number of encodings in the original obfuscation, to m^d, the number of all possible monomials. While such a polynomial blowup is acceptable in the context of IO, for XIO with compression $d^{-1} \leq \gamma < 1$, it is devastating.

Key Idea: XIO in Decomposable Form. To overcome the above difficulty, we observe that the known black-box construction of XIO from FE [BNPW16] has certain structural properties that we can exploit. At a very high level, it can be decomposed into smaller pieces, so that instead of computing *all* monomials over *all* the encodings created during obfuscation, we only need to consider a much smaller subset of monomials. In this subset, each monomial only depends on a few small pieces, and thus only on few encodings.

To be more concrete, we next give a high-level account of this construction. To convey the idea in a simple setting of parameters, let us assume that we have at our disposal an FE scheme that support an unbounded number of keys, rather than a single key scheme, with the guarantee that the size of ciphertexts does not grow with the number of keys. In this case, the XIO scheme in [BNPW16] works as follows:

- To obfuscate a circuit C with n input bits, the scheme publishes a collection of function keys $\{\mathsf{SK}_{D_\tau}\}_\tau$ for circuits D_τ, indexed by prefixes $\tau \in \{0,1\}^{n/2}$ (will be specified shortly), and a collection of ciphertexts $\left\{\mathsf{CT}_{\rho\|C}\right\}_\rho$, each encrypting the circuit C and a suffix $\rho \in \{0,1\}^{n/2}$.
- Decrypting a ciphertext $\mathsf{CT}_{\rho\|C}$ with key SK_{D_τ} reveals $D_\tau(\rho\|C) := C(\tau, \rho)$.

The obfuscated circuit indeed has weakly subexponential size. It contains:

- $2^{n/2}$ function keys SK_{D_τ}, each of size $\mathrm{poly}(|C|)$,
- $2^{n/2}$ ciphertexts $\mathsf{CT}_{\rho\|C}$, each of size $\mathrm{poly}(|C|)$.

Going back to the ideal graded-encoding model, the FE key generation and encryption algorithms use the ideal oracle to encode elements. Therefore, generating the obfuscation involves generating a set of k encodings $\boldsymbol{h}_\tau = \{h_{\tau,i}\}_{i\in[k]}$ for each secret key SK_{D_τ} and a set of k encodings $\boldsymbol{h}_\rho = \{h_{\rho,i}\}_{i\in[k]}$ for each ciphertext $\mathsf{CT}_{\rho\|C}$, for some $k = \mathrm{poly}(|C|)$. The crucial point is that now, evaluating the obfuscation on a given input (τ, ρ) only involves the two small sets of encodings $\boldsymbol{h}_\tau, \boldsymbol{h}_\rho$. In particular, any zero-test made by the decryption algorithm is a polynomial defined only over the underlying field elements $\boldsymbol{\xi}_\tau = \{\xi_{\tau,i}\}_{i\in[k]}$ and $\boldsymbol{\xi}_\rho = \{\xi_{\rho,i}\}_{i\in[k]}$.

This gives rise to the following degree reduction strategy. In the obfuscation, rather then precomputing all monomials in all encodings as before, we precompute only the monomials corresponding to the different pieces $\left\{\Phi(\boldsymbol{\xi}_\rho)\right\}_{\rho,\Phi}, \{\Phi(\boldsymbol{\xi}_\tau)\}_{\tau,\Phi}$. Now, rather than representing zero-tests made by the decryption algorithm as linear polynomials in these monomials, they can be represented as quadratic polynomials

$$p(\boldsymbol{\xi}_\tau, \boldsymbol{\xi}_\rho) = Q_p\left(\{\Phi(\boldsymbol{\xi}_\tau)\}_\Phi, \left\{\Phi(\boldsymbol{\xi}_\rho)\right\}_\Phi\right) \ .$$

To support such quadratic zero tests, we resort to bilinear groups. We use the bilinear encoding oracle to encode the values $\left\{\Phi(\boldsymbol{\xi}_\rho)\right\}_{\rho,\Phi}, \{\Phi(\boldsymbol{\xi}_\tau)\}_{\tau,\Phi}$, and publish the corresponding handles $\{h_{\tau,\Phi}\}_{\tau,\Phi}, \{h_{\rho,\Phi}\}_{\rho,\Phi}$. Evaluation is done in a straight forward manner by testing the quadratic polynomial Q_p.

The key gain of this construction is that now the blowup is tolerable. Now, each set of k encodings, blows up to k^d, which is acceptable since $k = \mathrm{poly}(|C|)$ is small (and not proportional to the size of the entire obfuscation as before, which is exponential in n). In the body, we formulate a general *product form* property for XIO schemes, which can be used as the starting point of the above-described transformation; we further show that single-key FE schemes with $\frac{1}{d+\varepsilon}$-succinctness implies such XIO schemes.

A Closer Look. The above exposition is oversimplified. To actually fulfill our strategy, we need to overcome two main challenges.

Challenge 1: Explicit Handles. The core idea described above assumes that the obfuscation is simply given as an explicit list of handles, which may not be the case starting from an arbitrary FE scheme. In particular, the obfuscator may use the oracle $\mathcal{M}$ to produce a set of encodings, but not output them explicitly; indeed, it can output an arbitrary string. In this case, we can no longer apply the degree reduction technique, since we do not know which encodings are actually contained in the obfuscation. Naïvely publishing all monomials in all field elements ever encoded by the obfuscator may be insecure — some of these encodings, which are never explicitly included in the obfuscation, may leak information.

To handle XIO schemes constructed from general FE schemes, we need a way to make any "implicit" handles explicit, without compromising security. Our idea is to *learn* the *significant handles* that would later suffice for evaluation on most inputs, and publish them explicitly. This idea is inspired by [CKP15, PS16, MMN16] and their observation (already mentioned above) that in obfuscation, the evaluator's view, including all the handles it sees, is publicly and efficiently samplable. Roughly speaking, the learning process involves evaluating the obfuscated circuit on many random inputs and making explicit all handles involved in these evaluations. This learning process is done in a careful manner that preserves the local structure of the construction in [BNPW16] (namely, the product form).

The scheme resulting from the above learning process is only approximately correct — the obfuscation with explicit handles errs on say 10% of the inputs. We show that even such *approximate XIO* is sufficient for obtaining FE and IO in the plain model (this step is described later in this overview).

Challenge 2: Invalid Monomials. Another main challenge is that it may be insecure to publish encodings of all the monomials $\left\{\Phi(\boldsymbol{\xi}_\rho)\right\}_{\rho,\Phi}$, $\{\Phi(\boldsymbol{\xi}_\tau)\}_{\tau,\Phi}$. The problem is that some products $\Phi(\boldsymbol{\xi}_\rho)\cdot\Phi'(\boldsymbol{\xi}_\tau)$ may result in monomials that would have been invalid in the degree-d ideal model. For example, $\Phi(\boldsymbol{\xi}_\rho)$ could correspond to a degree-$(d-2)$ monomial Φ. In the degree-d ideal model, it would only be possible to multiply such a monomial by degree-2 monomials $\Phi'(\boldsymbol{\xi}_\tau)$, and zero test. In the the described new scheme, however, it can multiply monomials $\Phi'(\boldsymbol{\xi}_\tau)$ of degree 3, or even d, which might compromise security.

Our solution proceeds in two steps. First, we show how to properly preserve validity by going to a more structured model of bilinear encodings that generalizes *asymmetric* bilinear groups. In this model, every encoding contains one of many labels and only pairs of encodings with valid labels can be multiplied. We then encode the monomials $\left\{\Phi(\boldsymbol{\xi}_\rho)\right\}_{\rho,\Phi}$, $\{\Phi(\boldsymbol{\xi}_\tau)\}_{\tau,\Phi}$ with appropriate labels that preserve the information regarding the original set of labels. This guarantees that the set of monomials that can be zero-tested in this model corresponds exactly to the set of valid monomials in the constant-degree graded encoding model we started from.

Second, we show how to transform any construction in this (more structured) ideal model into one in the standard ideal bilinear encoding model (corresponding to *symmetric* bilinear maps). At a very high-level, we develop a "secret-key transformation" from asymmetric bilinear groups to symmetric bilinear groups. The transformation allows anyone in the possession of a secret key to translate encodings in the asymmetric setting to new encodings in the symmetric setting in a manner that enforces the asymmetric structure.

From Approximately-Correct XIO back to FE. After applying all the above steps, we obtain an approximately-correct XIO scheme in the ideal bilinear encoding model. The only remaining step is going from such an XIO scheme back to FE. The work of [LPST16b] showed how to construct FE from XIO with *perfect* correctness, assuming in addition LWE. We modify their transformation to construct FE starting directly from approximately-correct XIO. This is done using appropriate Error Correcting Codes to accommodate for the correctness errors from XIO. (We note that existing transformations for removing errors from IO

[BV16] do not work for XIO. See Section 7.3 for details.) The transformation uses XIO as a black-box, and can thus be performed in the ideal bilinear model.

Putting It All Together. Putting all pieces together, we finally obtain our transformation from $\frac{1}{d+\varepsilon}$-succinct FE in the constant-degree graded encoding model to weakly-succinct FE in the bilinear encoding model. To recap the structure of the transformation:

1. Start with a $\frac{1}{d+\varepsilon}$-succinct (single-key) FE in the ideal constant-degree graded encoding model.
2. Transform it into an XIO scheme in the ideal constant-degree graded encoding model satisfying an appropriate decomposition property (which we call product form).
3. Transform it into an approximate XIO scheme in the ideal bilinear encoding model.
4. Use the resulting approximate XIO scheme and LWE to get a weakly-succinct FE (still, in the ideal bilinear encoding model).

Instantiating the oracle in bilinear groups with über security gives a corresponding construction in the plain model.

2 Preliminaries

We use the following conventions:

- We say that a (uniform) Turing machine is PPT if it is probabilistic and runs in polynomial time.
- $\mathcal{C} = \{\mathcal{C}_\lambda\}_\lambda$ is said to be a class of polynomial-size circuits if each $\mathcal{C}_\lambda$ is a set of circuits of polynomial size $\lambda^{O(1)}$. We will also discuss collections of circuit classes $\boldsymbol{\mathcal{C}} = \{\mathcal{C}\}$, such as $\mathbf{P/poly}$, the collection of all classes of polynomial-size circuits, or $\mathbf{NC}^1$, the collection of all classes of polynomial size circuits and logarithmic depth. We shall sometimes abuse notation and for a circuit C may write $C \in \boldsymbol{\mathcal{C}}$ to denote the fact that $C \in \mathcal{C}_\lambda$ for some $\lambda \in \mathbb{N}$, and some $\mathcal{C} = \{\mathcal{C}_\lambda\} \in \boldsymbol{\mathcal{C}}$.
- We follow the standard habit of modeling any efficient adversary strategy as a family of polynomial-size circuits. For an adversary $\mathcal{A}$ corresponding to a family of polynomial-size circuits $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, we often omit the subscript λ, when it is clear from the context.
- We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for all constants $c > 0$, there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$.
- If $\mathcal{X}^{(b)} = \{X_\lambda^{(b)}\}_{\lambda\in\mathbb{N}}$ for $b \in \{0, 1\}$ are two ensembles of random variables indexed by $\lambda \in \mathbb{N}$, we say that $\mathcal{X}^{(0)}$ and $\mathcal{X}^{(1)}$ are computationally indistinguishable if for all polynomial-size distinguishers $\mathcal{D}$, there exists a negligible function ν such that for all λ,

$$\Delta = \left|\Pr[\mathcal{D}(X_\lambda^{(0)}) = 1] - \Pr[\mathcal{D}(X_\lambda^{(1)}) = 1]\right| \le \nu(\lambda).$$

- For a concrete negligible function $\delta(\lambda)$, we write $\mathcal{X}^{(0)} \approx_\delta \mathcal{X}^{(1)}$ to denote that the advantage Δ is bounded by $\delta(\lambda)^{\Omega(1)}$.

Fact 2.1 (Geometric Probability Bound)**.** *For any $\alpha < 1, K \in \mathbb{N}$, let $p := (1 - \alpha)^K \alpha$. Then*

1. $p \leq 1/K$,

2. *if* α *is the probability of some event over a uniformly random* X *from a finite set* $\mathcal{X}$, $p \leq 2^{-\frac{K}{|\mathcal{X}|}}$.

Fact 2.2 (Schwartz-Zippel)**.** *Let* $P_1, \dots, P_t, Q_1, \dots, Q_t \in \mathbb{F}[x_1, \dots, x_k]$ *be non-trivial polynomials over a finite field* $\mathbb{F}$ *each of total degree at most* d*. Then*

- **Simple Version:**

$$\Pr_{x_1,\dots,x_k \leftarrow \mathbb{F}}[P_1(x_1, \dots, x_k) = 0] \leq \frac{d}{|\mathbb{F}|} \quad ,$$

- **Extended Version:** *if* $\frac{P_1}{Q_1}, \dots, \frac{P_t}{Q_t}$ *are linearly independent, and* $\rho_1, \dots, \rho_t \in \mathbb{F} \setminus \{0\}$,

$$\Pr_{x_1,\dots,x_k \leftarrow \mathbb{F}}\left[\sum_{i \in [t]} \rho_i \frac{P_i(x_1, \dots, x_k)}{Q_i(x_1, \dots, x_k)} = 0\right] \leq \frac{dt}{|\mathbb{F}|} \quad .$$

Multi-sets. For two multi-sets $A = \{a_1, \dots, a_n\}$ and $B = \{b_1, \dots, b_m\}$, their union with multiplicity is the multi-set $A \uplus B = \{a_1, \dots, a_n, b_1, \dots, b_m\}$. For example, $\{1, 1\} \uplus \{1, 2\} = \{1, 1, 1, 2\}$.

2.1 XIO

Lin, Pass, Seth, and Telang [LPST16a] propose a variant of IO that has a weak (yet non-trivial) efficiency, which they call exponentially-efficient IO (XIO). All that this notion requires in terms of efficiency is that the size of an obfuscated circuit is weakly sublinear in the size of the corresponding truth table. Here we also define *approximately-correct* XIO, which is only required to preserve functionality on some large enough fraction of inputs. We next formally define the notion of exponentially-efficient indistinguishability obfuscation (XIO) for any collection of circuit classes $\boldsymbol{\mathcal{C}} \subseteq \mathbf{P}^{\log}/\mathbf{poly}$, where $\mathbf{P}^{\log}/\mathbf{poly}$ is the collection of all classes of polynomial-size circuits with logarithmic size input.

Definition 2.1 ($\mathbf{P}^{\log}/\mathbf{poly}$)**.** *The collection* $\mathbf{P}^{\log}/\mathbf{poly}$ *includes all classes* $\mathcal{C} = \{\mathcal{C}_\lambda\}$ *for which there exists a constant* $c = c(\mathcal{C})$*, such that the input of any circuit* $C \in \mathcal{C}_\lambda$ *is bounded by* $c \log \lambda$ *and the size of* C *is bounded by* λ^c.

Definition 2.2 (XIO [LPST16a])**.** *A pair of algorithms* $\mathsf{xiO} = (\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ *is an* exponentially-efficient indistinguishability obfuscator *(XIO) for a collection of circuit classes* $\boldsymbol{\mathcal{C}} = \{\mathcal{C} = \{\mathcal{C}_\lambda\}\} \subseteq \mathbf{P}^{\log}/\mathbf{poly}$ *if it satisfies:*

- ***Functionality:*** *for any* $\mathcal{C} \in \boldsymbol{\mathcal{C}}$*, security parameter* $\lambda \in \mathbb{N}$*, and* $C \in \mathcal{C}_\lambda$ *with input size* n*,*

$$\Pr_{\substack{\mathsf{xiO} \\ x \leftarrow \{0,1\}^n}}[\mathsf{xiO.Eval}(\widetilde{C}, x) = C(x) : \widetilde{C} \leftarrow \mathsf{xiO.Obf}(C, 1^\lambda)] \geq 1 - \alpha(\lambda) \quad .$$

We say that $\mathsf{xiO.Obf}$ *is correct if* $\alpha(\lambda) \leq \mathrm{negl}(\lambda)$ *and approximately-correct if* $\alpha(\lambda) \leq 1/100$.

- ***Non-Trivial Efficiency:*** *there exists a constant* $\gamma < 1$ *and a fixed polynomial* $\mathrm{poly}(\cdot)$*, depending on the collection* $\boldsymbol{\mathcal{C}}$ *(but not on any specific class* $\mathcal{C} \in \boldsymbol{\mathcal{C}}$*), such that for any class* $\mathcal{C} \in \boldsymbol{\mathcal{C}}$ *security parameter* $\lambda \in \mathbb{N}$*, circuit* $C \in \mathcal{C}_\lambda$ *with input length* n*, and input* $x \in \{0,1\}^n$ *the running time of both* $\mathsf{xiO.Obf}(C, 1^\lambda)$ *and* $\mathsf{xiO.Eval}(\widetilde{C}, x)$ *is at most* $\mathrm{poly}(2^n, \lambda, |C|)$ *and the size of the obfuscated circuit* $\widetilde{C}$ *is at most* $2^{n\gamma} \cdot \mathrm{poly}(|C|, \lambda)$*. We call* γ *the compression factor, and say that the scheme is* γ*-compressing.*

- ***Indistinguishability:*** *for any* $\mathcal{C} = \{\mathcal{C}_\lambda\} \in \boldsymbol{\mathcal{C}}$ *and polynomial-size distinguisher* $\mathcal{D}$*, there exists a negligible function* $\mu(\cdot)$ *such that the following holds: for all security parameters* $\lambda \in \mathbb{N}$*, for any pair of circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *of the same size and such that* $C_0(x) = C_1(x)$ *for all inputs* x*,*

$$\left|\Pr\left[\mathcal{D}(\mathsf{xiO.Obf}(C_0, 1^\lambda)) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{xiO.Obf}(C_1, 1^\lambda)) = 1\right]\right| \leq \mu(\lambda) \ .$$

We further say that $\mathsf{xiO.Obf}$ *is* δ*-secure, for some concrete negligible function* $\delta(\cdot)$*, if for all polynomial-size distinguishers the above indistinguishability gap* $\mu(\lambda)$ *is smaller than* $\delta(\lambda)^{\Omega(1)}$*.*

Remark 2.1 (Logarithmic Input). Indeed, for XIO to be useful, we must restrict attention to circuit collections $\boldsymbol{\mathcal{C}} \subseteq \mathbf{P}^{\log}/\mathbf{poly}$. This ensures that obfuscation and evaluation time, which depend on $2^{O(n)} = \mathrm{poly}(\lambda)$, are polynomial in the security parameter λ.

Remark 2.2 (Probabilistic $\mathsf{xiO.Eval}$). Above, we allow the evaluation algorithm $\mathsf{xiO.Eval}$ to be probabilistic. Throughout most of the paper, and unless stated otherwise (Section 6), we restrict attention to *deterministic* evaluation algorithms. This typically will simplify exposition and is without loss of generality. Indeed, we can always let the obfuscation algorithm output, together with the obfuscation, coins $r \leftarrow \{0,1\}^\lambda$ to be used by the evaluation algorithm (when needed, these can be expanded to a pseudo random string of any polynomial size $\mathrm{poly}(\lambda)$).

XIO with an Oracle. We say that an XIO scheme $\mathsf{xiO} = (\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ is constructed relative to an oracle $\mathcal{O}$ if the corresponding algorithms, as well as the adversary, may access the oracle $\mathcal{O}$. Namely, the obfuscation algorithm $\mathsf{xiO.Obf}^{\mathcal{O}}(C, 1^\lambda)$ and the evaluation algorithm $\mathsf{xiO.Eval}^{\mathcal{O}}(\widetilde{C}, x)$ are given oracle access to $\mathcal{O}$. In the security definition, the adversarial distinguisher $\mathcal{D}^{\mathcal{O}}$ also gets access to the oracle.

When discussing XIO schemes relative to oracles we will address two query-complexity measures:

- The *obfuscation query complexity* denoted by $q_o = q_o(C, \lambda)$ is a bound on the total size $\sum_{Q \in \mathbf{Q}_o} |Q|$ of oracle queries $\mathbf{Q}_o = \{Q\}$ made by $\mathsf{xiO.Obf}^{\mathcal{O}}(C, 1^\lambda)$ when obfuscating a circuit C. (In particular, it bounds their number $|\mathbf{Q}_o|$.)
- The *evaluation query complexity* denoted by $q_e = q_e(C, \lambda)$ is a bound on the total size $\sum_{Q \in \mathbf{Q}_e} |Q|$ oracle queries $\mathbf{Q}_e = \{Q\}$ made by $\mathsf{xiO.Eval}^{\mathcal{O}}(\widetilde{C}, x)$ when evaluating an obfuscation $\widetilde{C}$ of a circuit C on input x.

2.2 Puncturable Pseudorandom Functions

Puncturable PRFs, defined by Sahai and Waters [SW14], are PRFs with a key-puncturing procedure that produces keys that allow evaluation of the PRF on all inputs, except for a designated point.

Definition 2.3 (Puncturable Pseudorandom Functions)**.** *For sets* $\mathcal{D}, \mathcal{R}$*, a* puncturable pseudorandom function *(PPRF) consists of a tuple of algorithms* $\mathcal{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$ *that satisfy the following two conditions.*

- **Functionality Preserving Puncturing:** *For any distinct* $x, x^* \in \mathcal{D}$*, it holds that*

$$\Pr\left[\mathsf{PRF.Eval}_K(x) = \mathsf{PRF.Eval}_{K\{x^*\}}(x) \;\middle|\; \begin{array}{l} K \leftarrow \mathsf{PRF.Gen}(1^\lambda) \\ K\{x^*\} \leftarrow \mathsf{PRF.Punc}(K, x^*) \end{array}\right] = 1 \ .$$

- **Pseudorandomness at Punctured Points:** *For any* $x^* \in \mathcal{D}$ *and any polynomial-size distinguisher* $\mathcal{A}$*, there exists a negligible function* μ*, such that:*

$$|\Pr[\mathcal{A}(\mathsf{PRF.Eval}_{K\{x^*\}}, \mathsf{PRF.Eval}_K(x^*)) = 1] - \Pr[\mathcal{A}(\mathsf{PRF.Eval}_{K\{x^*\}}, U) = 1]| \leq \mu(\lambda) \ ,$$

where $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$*,* $K\{x^*\} \leftarrow \mathsf{PRF.Punc}(K, x^*)$ *and* U *denotes the uniform distribution over* $\mathcal{R}$*. We further say that* $\mathcal{PPRF}$ *is* δ*-secure, for some concrete negligible function* $\delta(\cdot)$*, if for all polynomial-size distinguishers the above indistinguishability gap* $\mu(\lambda)$ *is smaller than* $\delta(\lambda)^{\Omega(1)}$*.*

The GGM tree-based construction of PRFs [GGM84] yield puncturable PRFs [BW13, BGI14, KPTZ13].

3 Oracles

We review the oracle models addressed in this work.

3.1 The Random-Oracle Model

In the random-oracle model [BR93], all algorithms get access to a random function.

Definition 3.1 (Random Oracle)**.** *A random oracle* $\mathcal{R} = \{\mathcal{R}_\lambda\}$*, for some polynomially-bounded function* ℓ*, is an ensemble of random functions* $\mathcal{R}_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$*.*

3.2 The Ideal Graded Encoding Model

The ideal graded-encoding model we consider is inspired by previous generic-group and ideal graded-encoding models [Sho97, Mau05, BR14, BGK+14] and is closest to the model of Pass and Shelat [PS16]. The model is given by an oracle $\mathcal{M}$ parameterized by a finite field $\mathbb{F}$. The oracle enables to *encode* any element $\xi \in \mathbb{F}$ under a label ℓ and obtain a representation of (ξ, ℓ) as a handle $h = (r, \ell)$ consisting of a random string r and the corresponding label. We explicitly include the label ℓ in the handle to capture the fact that it may be public. In addition, the oracle allows *valid* zero-tests over encoded elements. A zero-test query is given by a list of handles $\boldsymbol{h} = (h_1, \ldots, h_m)$ and an m-variate polynomial $p(\boldsymbol{v})$ in formal variables $\boldsymbol{v} = (v_1, \ldots, v_m)$. The oracle obtains the corresponding field elements along with their labels $(\xi_1, \ell_1), \ldots, (\xi_m, \ell_m)$. It first checks whether the zero test is valid by applying a validity predicate $V(p, \ell_1, \ldots, \ell_m)$, and if so it tests whether $p(\xi_1, \ldots, \xi_m) = 0$. As in [PS16], we consider well-formed predicates that are determined by the validity of monomials.

Definition 3.2 (Well-Formed Validity Predicate)**.** V *is well-formed if for any* $d \in \mathbb{N}$ *and degree-*d *polynomial*

$$p(v_1, \ldots, v_m) = \sum_{i \leq d, j_1, \ldots, j_i \in [m]} \rho_{j_1,\ldots,j_i} v_{j_1} \cdots v_{j_i} \ ,$$

it holds that

$$V(p, \ell_1, \ldots, \ell_m) = \bigwedge_{\substack{i \leq d, j_1, \ldots, j_i \in [m] \\ \rho_{j_1,\ldots,j_i} \neq 0}} V(\{\ell_{j_1}, \ldots, \ell_{j_i}\}) \ ;$$

namely, p *is valid relative to the labels* $\ell_1, \ldots, \ell_m$ *if every monomial of* p *is valid relative to the corresponding multi-set of labels* $\{\ell_{j_1}, \ldots, \ell_{j_i}\}$*.*

In addition to well-formedness, we consider a decomposability requirement. Specifically, a *decomposable* validity predicate $V(\{\ell_1, \cdots, \ell_k\})$ is defined by a simpler two-input validity predicate V_Π and a projection function Π in the following way: For any two multi-sets of labels A, B, the validity $V(A \uplus B)$ of their union (with multiplicity) $A \uplus B$ is determined by $V_\Pi(\Pi(A), \Pi(B))$.

Definition 3.3 (Decomposable Validity Predicate)**.** *V is* decomposable *if it is well-formed and there exist a projection function Π and a two-input predicate V_Π satisfying: For every two multi-sets A and B of labels, the validity*

$$V(A \uplus B) = V_\Pi(\Pi(A), \Pi(B)) \ .$$

The arity *of a decomposable predicate V is*

$$\mathsf{Arity}(V) := \max_A |\{\Pi(B) \ : \ V_\Pi(\Pi(A), \Pi(B)) = 1 \ \}| \ \ ;$$

namely, it is the maximum number of projections $\Pi(B)$ that satisfy the validity predicate together with any given projection $\Pi(A)$, where A and B are multi-sets of labels.

Intuitively, for decomposable validity predicates, the only information about multi-sets that matters for validity is their projection. In the literature, all known ideal graded encoding models consider decomposable validity predicates with arity bounded by the degree (or even less). For instance, in set-based graded encodings, the labels correspond to subsets of some fixed universe $\mathbb{U}$, and a set of labels $\{S_1, \ldots, S_k \mid S_i \subseteq \mathbb{U}\}$ is valid if the sets are disjoint and $\biguplus S_i = \mathbb{U}$. Therefore, we can define the projection of any $A = \{S_1, \ldots, S_n\}$ to be $\Pi(A) = \biguplus S_i$ (or $\perp$ if the sets are not disjoint). In this case the arity is exactly one; indeed, for any A and $\Pi(A)$ only sets $B = \{T_1, \ldots, T_m\}$ such that $\Pi(B) = \biguplus T_i = \mathbb{U} \setminus \Pi(A)$ are valid.

Similarly, in integer-based graded encodings, validity only depends on the sum of the integer-labels in a multi-set. This sum can be thought of as the required projection, and the arity is bounded by the number of levels in which zero-testing is allowed, which is in turn bounded by the degree. In graph-based encodings, the arity is upper bounded by the total number of pathes in the graph, which is some fixed polynomial.

We remark that the property of decomposability with bounded arity will be instrumental in our transformation later (in Section 4.3).

We now formally define the ideal graded encoding model.

Definition 3.4 (Ideal Graded Encoding Oracle)**.** *The oracle $\mathcal{M}_{\mathbb{F},V}$ is a stateful oracle, parameterized by a field $\mathbb{F}$ and a validity predicate V. The oracle answers queries of two forms:*

1. **Encoding Queries:** *Given a field element $\xi \in \mathbb{F}$ and label ℓ, the oracle samples a uniformly random string $r \leftarrow \{0,1\}^{\log|\mathbb{F}|}$, returns the handle $h = (r, \ell)$, and stores (h, ξ).*
2. **Zero-Test Queries:** *Given a polynomial $p \in \mathbb{F}[v_1, \ldots, v_m]$, and handles $h_1, \ldots, h_m$, the oracle does the following:*
   - *For each $i \in [m]$, obtains a tuple (h_i, ξ_i) from the stored list. If no such tuple exists, stops and returns* `false` .
   - *From each $h_i = (r_i, \ell_i)$, obtains ℓ_i, and checks that $V(p, \ell_1, \ldots, \ell_m) =$* `true` *to verify the query is valid and if not, returns* `false` .
   - *Performs a zero test, returning* `true` *if $p(\xi_1, \ldots, \xi_m) = 0$ and* `false` *otherwise.*

An ideal graded encoding oracle $\mathcal{M} = \{\mathcal{M}_{\mathbb{F}_\lambda, V_\lambda}\}$ is a collection of oracles $\mathcal{M}_{\mathbb{F}_\lambda, V_\lambda}$, one for each $\lambda \in \mathbb{N}$, where $|\mathbb{F}_\lambda| = 2^{\Theta(\lambda)}$.

Additionally:

- *The oracle $\mathcal{M}$ is said to be degree-d, if for every polynomial p of degree $\deg(p) > d$, and any label vector $\boldsymbol{\ell}$, $V(p, \boldsymbol{\ell}) = \texttt{false}$.*
- *The oracle $\mathcal{M}$ is said to be decomposable if it has a decomposable validity predicate with bounded polynomial arity $\mathsf{Arity}(V_\lambda) \leq \text{poly}(\lambda)$.*

Remark 3.1 (Public Encoding). In some previous models (e.g., [PS16]), the ability to make encoding queries is restricted. In the context of obfuscation, this is modeled by allowing only the obfuscator to *initialize* the oracle with encodings. More generally, it can be modeled by providing only relevant parties (like the obfuscator) with a *secret encoding key*. (For instance, one can think of secret-key functional encryption schemes where the encryptor and key generator require this secret key, but functional decryption does not.)

The above definition does not enforce this restriction. The results in the paper are presented in the model of public encodings. The results in Section 5 extend also to the model of private encodings. The results in Section 4, extend to the model of private encodings assuming either: (a) it is possible to publicly generate encodings of random strings (which is indeed, the case in existing candidate graded encoding schemes), or (b) "zero-testing in low levels is allowed"; namely, the validity predicate V is such that if $V(L) = \texttt{true}$, then for any $L' \subseteq L$, $V(L') = \texttt{true}$.

Remark 3.2 (Beyond Fields). The model considered here restricts attention to fields $\mathbb{F}$. The results naturally extends also to *pseudo-fields* such as the ring $\mathbb{Z}_N$ for composite N, assuming factoring N is hard.

4 Reducing Constant-Degree Oracles to Bilinear Oracles

We show that any XIO scheme with a constant-degree decomposable ideal oracle can be transformed into an approximately-correct one with an ideal symmetric bilinear oracle (analogous to symmetric bilinear groups), provided that the XIO scheme is in a certain *product form*. In Section 7.2, we show that any single-key functional encryption scheme with a constant-degree decomposable ideal oracle, and appropriate succinctness, implies such an XIO scheme. We start by defining the notion of XIO in product form and of a symmetric bilinear oracle, then we give a high-level description of the transformation, and move on to describe the transformation in more detail.

XIO in Product Form. Roughly speaking, an XIO scheme in *product form* is associated with a *product collection* of sets $\mathcal{X} = \{X\}, \mathcal{Y} = \{Y\}$ that factors the inputs space

$$\{0,1\}^n \cong \biguplus_{X \in \mathcal{X}} X \times \biguplus_{Y \in \mathcal{Y}} Y$$

so that obfuscation and evaluation decompose accordingly into pieces:

- An obfuscation $\widetilde{C}$ of a circuit C consists of $|\mathcal{X}| + |\mathcal{Y}|$ pieces $\left\{\widetilde{C}_X \;\middle|\; X \in \mathcal{X}\right\}$ and $\left\{\widetilde{C}_Y \;\middle|\; Y \in \mathcal{Y}\right\}$.
- Given pieces $\widetilde{C}_X, \widetilde{C}_Y$, one can evaluate the obfuscation, namely compute $C(x, y)$, for all $(x, y) \in X \times Y$.

The main advantage in product form is that, when the sets X, Y in the collection are small, evaluation at a given point (x, y) may potentially be a local operation that depends only on a small part of the entire XIO obfuscation. We next formally define the notions of a product collection and XIO in product form.

Definition 4.1 (Product Collection)**.** *$\mathcal{X} = \{\mathcal{X}_n\}_{n\in\mathbb{N}}, \mathcal{Y} = \{\mathcal{Y}_n\}_{n\in\mathbb{N}}$ are said to be a product collection if:*

1. **Equal-Size Partition:** *For any $X, X' \in \mathcal{X}_n$ and $Y, Y' \in \mathcal{Y}_n$:*

$$|X| = |X'|, X \cap X' = \emptyset \qquad |Y| = |Y'|, Y \cap Y' = \emptyset \ ,$$

2. **Product Form:** *let $\boldsymbol{X}_n = \biguplus_{X\in\mathcal{X}_n} X, \boldsymbol{Y}_n = \biguplus_{Y\in\mathcal{Y}_n} Y$ then the input space $\{0,1\}^n$ factors:*

$$\{0,1\}^n \cong \boldsymbol{X}_n \times \boldsymbol{Y}_n \ .$$

Definition 4.2 (XIO in Product Form)**.** *We say that an XIO scheme $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{O}}, \mathsf{xiO.Eval}^{\mathcal{O}})$, relative to oracle $\mathcal{O}$, for a collection of circuit classes $\boldsymbol{\mathcal{C}}$, is in $(\mathcal{X}, \mathcal{Y})$-product form for a product collection $(\mathcal{X}, \mathcal{Y})$ if:*

- *The obfuscation algorithm $\mathsf{xiO.Obf}^{\mathcal{O}}$ factors into two algorithms $(\mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{X}}, \mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{Y}})$, such that for any circuit $C \in \boldsymbol{\mathcal{C}}$, The obfuscation algorithm $\mathsf{xiO.Obf}^{\mathcal{O}}(C, 1^\lambda; r)$ outputs*

$$\left(\left\{\widetilde{C}_X \leftarrow \mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{X}}(C, X, 1^\lambda; r)\right\}_{X\in\mathcal{X}_n}, \left\{\widetilde{C}_Y \leftarrow \mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{Y}}(C, Y, 1^\lambda; r)\right\}_{Y\in\mathcal{Y}_n}\right) \ ,$$

  and all executions may use joint randomness r.

- *There is an evaluation algorithm $\mathsf{xiO.Eval}^{\mathcal{O}}_{\mathcal{X},\mathcal{Y}}$ such that for any $(X, Y) \in \mathcal{X}_n \times \mathcal{Y}_n$,*

$$\mathsf{xiO.Eval}^{\mathcal{O}}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y) = \left(\mathsf{xiO.Eval}^{\mathcal{O}}(\widetilde{C}, (x, y))\right)_{(x,y)\in X\times Y} \ .$$

Corresponding notation:

- *We denote by $q^{\mathcal{X}}_o = q^{\mathcal{X}}_o(C, \lambda)$ the maximal total size $\max_X \sum_{Q\in\mathbf{Q}^X_o} |Q|$ of all oracle queries $\mathbf{Q}^X_o = \{Q\}$ made by $\mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{X}}(C, X, 1^\lambda)$ when obfuscating an n-bit input circuit $C \in \boldsymbol{\mathcal{C}}$ for any $X \in \mathcal{X}_n$. Symmetrically, we denote by $q^{\mathcal{Y}}_o = q^{\mathcal{Y}}_o(C, \lambda)$ the bound on the total size $\max_Y \sum_{Q\in\mathbf{Q}^Y_o} |Q|$ of oracle queries $\mathbf{Q}^Y_o = \{Q\}$ made by $\mathsf{xiO.Obf}^{\mathcal{O}}_{\mathcal{Y}}(C, Y, 1^\lambda)$ for any $Y \in \mathcal{Y}_n$.*

The Bilinear Symmetric Oracle. The symmetric bilinear oracle is a special case of a degree-2 graded encoding oracle that is analogous to the symmetric bilinear model where there is a single base group G with a bilinear map $e : G \times G \to G_T$. In our terms, one can only encode field elements with respect to a single label (representing a single base group).

Definition 4.3 (Symmetric Bilinear Oracle)**.** *The symmetric Bilinear Oracle $\mathcal{B}^2 = \left\{\mathcal{B}^2_{\mathbb{F}_\lambda, V_\lambda}\right\}$ is a special case of the ideal graded encoding oracle, where the validity predicate V is of degree two and is defined over a single label $\ell_{\mathcal{B}}$. That is, $V(L) = \mathtt{true}$ for a multi-set of labels L, if and only if $L \subseteq \{\ell_{\mathcal{B}}, \ell_{\mathcal{B}}\}$.*

We now state the main theorem of this section.

Theorem 4.1. *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{M}}, \mathsf{xiO.Eval}^{\mathcal{M}})$ *be an* $\mathsf{xiO.Obf}$ *scheme, relative to a degree-d decomposable ideal graded encoding oracle $\mathcal{M}$, for a collection of circuit classes $\boldsymbol{\mathcal{C}}$ that is in $(\mathcal{X}, \mathcal{Y})$-product form, for some product collection $(\mathcal{X}, \mathcal{Y})$. Further assume that for some constant $\gamma < 1$,*

$$|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}} \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right)\right)^d + |\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}} \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right)\right)^d \le 2^{\gamma n} \cdot \mathrm{poly}(|C|, \lambda) \ .$$

Then xiO *can be converted into an approximately-correct scheme* $\mathsf{xiO}^\star$ *relative to the symmetric bilinear oracle $\mathcal{B}^2$.*

Remark 4.1. A slightly easier to parse version of the above condition, with some loss in parameters, is that

$$|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}}\right)^{2d} + |\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}}\right)^{2d} \le 2^{\gamma n} \cdot \mathrm{poly}(|C|, \lambda) \ .$$

Remark 4.2. Our ideal symmetric bilinear oracle captures symmetric bilinear pairing groups, but with two small gaps: Our oracle generates randomized encodings (following the Pass-shelat model) whereas bilinear pairing groups have unique encodings (of the form g^a), and our oracle does not support homomorphic opeartions whereas bilinear paring groups do. These differences are not consequential. In Section 4.4, we show how to instantiate the transformed XIO schemes produced by the above theorem using concrete bilinear pairing groups.

Without Loss of Generality. Throughout this section, we make the following assumptions without loss of generality. In the model of private encodings (Remark 3.1), the same assumptions can be made (see for instance [PS16]).

- **Obfuscator only encodes:** The XIO obfuscation algorithm only performs encoding queries and does not perform any zero tests. This is without loss of generality, as the obfuscator knows the field elements and labels underlying any generated handle (it encoded them itself), so zero-tests can be internally simulated.

- **Evaluator and adversary only zero-test:** The XIO evaluation algorithm as well as the adversary only perform zero tests and do not encode any elements themselves. Indeed, encoding of any (ξ, ℓ) can be internally simulated by sampling a corresponding handle $\widetilde{h}$. Then, whenever a zero-test $(p, h_1, \ldots, h_m, \widetilde{h}_1, \ldots, \widetilde{h}_{\widetilde{m}})$ includes such self-simulated handles $\widetilde{h}_i$, it is translated to a new zero test that does not include such handles, by hardwiring the required field elements into the polynomial p.

  In more detail, we may assume that any obfuscation always includes encodings of 1 under all labels $\ell \in \mathbb{L}$; this indeed is w.l.o.g in the public encoding model, as any adversary can generate such encodings by itself. We generate from $(p, h_1, \ldots, h_m, \widetilde{h}_1, \ldots, \widetilde{h}_{\widetilde{m}})$ a new zero-test $(p^\star, h_1, \ldots, h_m, h^1_{i_1}, \ldots, h^1_{i_{\widetilde{m}}})$, where $h^1_{i_j}$ is the 1-encoding under the label of handle $\widetilde{h}_j$. The polynomial $p^\star$ is defined to be $p^\star(h_1, \ldots, h_m, \widetilde{h}_1, \ldots, \widetilde{h}_{\widetilde{m}}) = p(h_1, \ldots, h_m, \xi_1 h^1_{i_1}, \ldots, \xi_{\widetilde{m}} h^1_{i_{\widetilde{m}}})$, where ξ_i are the underlying field elements for which encodings were self-simulated.

The High-Level Ideas and Structure of the Transformation. The high-level idea behind the transformation is as follows. Starting with an XIO in product form, imagine (for the time being) that the obfuscation $\widetilde{C}_X$ of any piece $X \in \mathcal{X}_n$ simply consists of a set of handles $\widetilde{H}_X = \{h_{X,1}, \ldots, h_{X,m_X}\}$ representing field elements encoded during obfuscation (Similarly for any $Y \in \mathcal{Y}_n$, we have a corresponding $\widetilde{H}_Y$). Then, by the product form, evaluating the obfuscation on any set of inputs (X, Y) only involves zero-tests of the form

$$p(h_{X,1}, \ldots, h_{X,m_X}, h_{Y,1}, \ldots, h_{Y,m_Y})$$

for some degree-d polynomial p over the formal variables $\widetilde{H}_X, \widetilde{H}_Y$. We can further factor p according to these two sets of variables and write

$$p(h_{X,1}, \ldots, h_{X,m_X}, h_{Y,1}, \ldots, h_{Y,m_Y}) = \sum_{i,j} \rho_{i,j} \Phi_i(h_{X,1}, \ldots, h_{X,m_X}) \Phi_j(h_{Y,1}, \ldots, h_{Y,m_Y}) \ ,$$

for some monomials Φ_i, Φ_j of total (joint) degree at most d.

The basic observation is that, from a functionality standpoint, *it is sufficient to have encodings (or rather, handles) of all possible degree-d monomials in each set $\widetilde{H}_X$ (respectively, $\widetilde{H}_Y$) and the ability to perform only degree-2 zero-tests over these encodings.* This of course will blowup the size of any $\widetilde{H}_X$ (or any $\widetilde{H}_Y$) exponentially in the degree d, $|\widetilde{H}_X|^d$ (or $|\widetilde{H}_Y|^d$). However, provided that these sets are small enough (e.g., polynomial in the security parameter), then this blowup does not foil the compression of the original XIO construction.

To fulfil this high-level idea, we need to deal with two main issues:

- Our assumption that $\widetilde{C}_X$ simply consists of a bunch of handles *is not without loss of generality*. Indeed, we want to deal with arbitrary obfuscators that may potentially have an arbitrary output, where the actual handles required for evaluation are not necessarily explicit.
- Even given all the explicit handles required for evaluation, $\widetilde{H}_X, \widetilde{H}_Y$ for all X, Y, encoding directly all possible monomials may possibly allow an attacker to zero-test polynomials it should not be able to. A simple example is that encoding two monomials Φ_i, Φ_j of degree d will allow the attacker to learn a monomial $\Phi_{i,j} = \Phi_i \cdot \Phi_j$ of degree $2d$, which it should not be able to.

Our actual transformation deals with these two issues and proceeds in three steps:

1. We first show how to transform an arbitrary XIO in product form into one where all handles needed for evaluation are explicitly given as part of the obfuscation. This step involves a certain probabilistic learning process, and results in an XIO with approximate correctness.
2. We then apply the *encode all monomials* approach described above, but using a more expressive bilinear oracle that supports a validity predicate that is engineered to match exactly the validity predicate of the constant-degree oracle that we start with. Intuitively, this model is an extension of the standard notion of asymmetric bilinear maps where instead of two base groups we may have a larger number of base groups $G_1, \ldots, G_n$ with a collection of *valid bilinear maps* $\{e_k : G_{i_k} \times G_{j_k} \to G_T\}$.
3. Finally, we give a way to simulate the above (extended) asymmetric bilinear model in the symmetric bilinear model. In this step, we rely on the fact that the ideal oracle has a decomposable validity predicate with bounded polynomial arity to simplify the label structure and ensure that the size of the obfuscated circuits only blows up by a fixed polynomial factor.

4.1 Step 1: Explicit Handles

In this section, we show how to transform any XIO in product form relative to an ideal degree-d oracle (not necessarily decomposable) into one where all handles required for evaluation are given explicitly (also in product form). We start by defining the notion of explicit handles in product form and then state and describe the transformation.

Definition 4.4 (Explicit Handles in Product Form)**.** *An XIO scheme* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{M}}, \mathsf{xiO.Eval}^{\mathcal{M}})$*, relative to an ideal graded encoding oracle, for a collection of circuit classes* $\boldsymbol{\mathcal{C}}$*, is said to have explicit handles in* $(\mathcal{X}, \mathcal{Y})$*-product form, for a product collection* $(\mathcal{X}, \mathcal{Y})$*, if the obfuscation and evaluation algorithms satisfy the following structural requirement:*

- *The obfuscation algorithm* $\mathsf{xiO.Obf}^{\mathcal{M}}(C, 1^\lambda)$ *outputs:*

$$\widetilde{C} = \left(\widetilde{Z}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}\right) \quad ,$$

  where each $\widetilde{H}_X$ *and* $\widetilde{H}_{\mathcal{Y}}$ *are sets of handles generated by the oracle* $\mathcal{M}$ *during obfuscation, and* $\widetilde{Z}$ *is arbitrary auxiliary information.*

- *All* true *zero-test queries* $(p, h_1, \ldots, h_m)$ *— that is, zero-test queries that evaluate to* `true` *— made by the evaluation algorithm* $\mathsf{xiO.Eval}^{\mathcal{M}}\left(\widetilde{C}, (x, y)\right)$ *are such that for all* $j \in [m]$*,* $h_j \in \widetilde{H}_X \cup \widetilde{H}_Y$*, where* $(X, Y) \in \mathcal{X}_n \times \mathcal{Y}_n$ *are the (unique) sets such that* $(x, y) \in X \times Y$*.*

Corresponding notation:

- *We denote by* $q_h^{\mathcal{X}} = q_h^{\mathcal{X}}(C, \lambda)$ *the bound* $\max_{X \in \mathcal{X}_n} |\widetilde{H}_X|$ *on the maximum size of the set of explicit handles corresponding to any* $X \in \mathcal{X}_n$*. We denote by* $q_h^{\mathcal{Y}} = q_h^{\mathcal{Y}}(C, \lambda)$ *the bound on* $\max_{Y \in \mathcal{Y}_n} |\widetilde{H}_Y|$*.*

We show that any $\mathsf{xiO.Obf}$ scheme relative to an ideal graded encoding oracle that is in product form can be turned into one that has explicit handles in product form, but is approximately correct.

Lemma 4.1. *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{M}}, \mathsf{xiO.Eval}^{\mathcal{M}})$ *be an* $\mathsf{xiO.Obf}$ *scheme, relative to an ideal graded encoding oracle* $\mathcal{M}$*, for a collection of circuit classes* $\boldsymbol{\mathcal{C}}$*, that is in* $(\mathcal{X}, \mathcal{Y})$*-product form, for some product collection* $(\mathcal{X}, \mathcal{Y})$*. Then* xiO *can be converted into a new approximately-correct scheme* $\mathsf{xiO}^\star$ *with explicit handles in* $(\mathcal{X}, \mathcal{Y})$*-product form relative to the same oracle* $\mathcal{M}$*.*

Furthermore, the size of the explicit handle sets are bounded as follows

$$q_h^{\mathcal{X}} \leq O\left(q_o^{\mathcal{X}}\right) \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right) \quad ,$$
$$q_h^{\mathcal{Y}} \leq O\left(q_o^{\mathcal{Y}}\right) \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right) \quad .$$

The High-Level Idea. The basic idea behind Lemma 4.1 is to augment the obfuscator so that in addition to obfuscating, it also tries to learn *the significant handles* that would suffice for evaluating the obfuscation on all, but perhaps a small fraction of, the inputs. For this purpose, inspired by techniques for removing ideal oracles from IO [CKP15, PS16, MMN16], we can evaluate the obfuscated program on many random inputs, and record all handles involved in the process. Doing this naïvely, however, does not preserve the product structure. Indeed, for our next step we would like that the explicit set of handles $\widetilde{H}$ that we learn should also be in product form $\{\widetilde{H}_X\}_{X \in \mathcal{X}_n}, \{\widetilde{H}_Y\}_{Y \in \mathcal{Y}_n}$ and satisfy the local evaluation requirement given by Definition 4.4.

To achieve the above goal, we need to perform a more careful learning process. Here a natural idea is to go through every $X \in \mathcal{X}_n$, and perform the same learning process, only restricted to this X. That is, evaluate the obfuscation for (X, Y_i) for many random sets $Y_i \in \mathcal{Y}_n$, so that we learn a significant set of handles $\widetilde{H}_X$ that would allow evaluating the obfuscation on (X, Y) for all, but a small fraction of, $Y \in \mathcal{Y}_n$. Then we could the same symmetrically for every $Y \in \mathcal{Y}$ to learn $\widetilde{H}_Y$. By averaging, for a random X, Y the set of handles $\widetilde{H}_X, \widetilde{H}_Y$ would include all relevant handles for evaluation.

The only thing we need to make sure is that the size of $\widetilde{H}_X, \widetilde{H}_Y$ is not too big, so that meaningful XIO compression is preserved. In the learning process just described, the size of these sets is affected by two quantities: the first quantity A is the number of random samples Y_i that are required to learn for each X, in order to get reasonable correctness. The second quantity B is the amount of handles, we add for every such sample. So that overall every $\widetilde{H}_X$ will be of size at most $A \cdot B$ (and the same symmetrically holds for every $\widetilde{H}_Y$). These quantities turn out to be controlled by the local complexity of obfuscation $q_o^{\mathcal{X}}, q_o^{\mathcal{Y}}$, and how they relate to each other. When the two are balanced $q_o^{\mathcal{X}} \approx q_o^{\mathcal{Y}}$, the learning process described above can indeed be to satisfy the bound given by Lemma 4.1 on the respective size of $\widetilde{H}_X, \widetilde{H}_Y$. For the general case, where possibly say $q_o^{\mathcal{X}} \gg q_o^{\mathcal{Y}}$, we need an even more careful learning process to achieve the required bound.

At a high level, the new learning process is not symmetric, it would start by learning for $\mathcal{X}$ (assuming $q_o^{\mathcal{X}} \gg q_o^{\mathcal{Y}}$) just as before. However, then when learning for $\mathcal{Y}$, it would perform an augment learning process that controls better the size of the resulting handle set h_Y. This is roughly done, by avoiding handles that were already learned during the previous learning phase for $\mathcal{X}$.

We now turn on to prove Lemma 4.1 formally. We first describe the new obfuscator in Section 4.1.1, then move on to analyze its parameter blowup in Section 4.1.2, approximate correctness in Section 4.1.3, and security in Section 4.1.4.

4.1.1 Our New XiO Scheme with Explicit Handles

The new obfuscator is constructed under the assumption that $q_o^{\mathcal{X}} \geq q_o^{\mathcal{Y}}$. This is w.l.o.g as otherwise, the obfuscator can reverse the roles of $\mathcal{X}, \mathcal{Y}$.

The Obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}$: Given a circuit $C \in \mathcal{C}$ with input-size n, and security parameter 1^λ, the obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}^{\mathcal{M}}(C, 1^\lambda)$ does the following:

- **Obfuscate:** Emulate the obfuscator

$$\mathsf{xiO.Obf}^{\mathcal{M}}(C, 1^\lambda) = \left(\left\{ \widetilde{C}_X \leftarrow \mathsf{xiO.Obf}^{\mathcal{M}}_{\mathcal{X}}(C, X, 1^\lambda) \right\}_{X \in \mathcal{X}_n}, \left\{ \widetilde{C}_Y \leftarrow \mathsf{xiO.Obf}^{\mathcal{M}}_{\mathcal{Y}}(C, Y, 1^\lambda) \right\}_{Y \in \mathcal{Y}_n} \right) .$$

  For each $X \in \mathcal{X}_n$ store a list L_X of all tuples (h, ξ) such that $\mathsf{xiO.Obf}^{\mathcal{M}}_{\mathcal{X}}(C, X, 1^\lambda)$ requested the oracle $\mathcal{M}$ to encode (ξ, ℓ) and obtained back a handle $h = (r, \ell)$. Store a similar list L_Y for each execution $\mathsf{xiO.Obf}^{\mathcal{M}}_{\mathcal{Y}}(C, Y, 1^\lambda)$.

- **Learn Heavy Handles for $\mathcal{X}_n$:** for each $X \in \mathcal{X}_n$, let $\widetilde{H}_X = \emptyset$.
  For $i \in \left\{1, \ldots, K_{\mathcal{X}} = \min\left(400 q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log\left(400 q_o^{\mathcal{X}}\right)\right)\right\}$ do:
  - Sample a random $Y_i \leftarrow \mathcal{Y}_n$.
  - Emulate $\mathsf{xiO.Eval}^{(\cdot)}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_{Y_i})$. To answer zero-test queries, emulate $\mathcal{M}$ using the lists (L_X, L_{Y_i}) constructed during the obfuscation phase. Given a zero test $(p, h_1, \ldots, h_m)$ that includes some handle $h_i \notin L_X \cup L_{Y_i}$, return `false` .
  - In the process, for every zero-test query $(p, h_1, \ldots, h_m)$, if $\mathcal{M}(p, h_1, \ldots, h_m) = \mathtt{true}$, namely it is a valid zero test and the answer is indeed zero, add $h_1, \ldots, h_m$ to $\widetilde{H}_X$.

  Store the resulting $\widetilde{H}_X$.

- **Learn Remaining Handles for $\mathcal{Y}_n$:** for each $Y \in \mathcal{Y}_n$, let $\widetilde{H}_Y = \emptyset$.
  For $i \in \left\{1, \ldots, K_{\mathcal{Y}} = \min\left(200 q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log\left(200 q_o^{\mathcal{Y}}\right)\right)\right\}$ do the following:

- Sample a random $X_i \leftarrow \mathcal{X}_n$, and let $\widetilde{H}_{X_i,Y} = \emptyset$.
- Emulate $\mathsf{xiO.Eval}^{(\cdot)}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_{X_i}, \widetilde{C}_Y)$. To answer zero-test queries, emulate $\mathcal{M}$ using the lists (L_{X_i}, L_Y) constructed during the obfuscation phase. Given a zero test $(p, h_1, \ldots, h_m)$ that includes some handle $h_i \notin L_{X_i} \cup L_Y$, return `false` .
- In the process, for every zero-test query $(p, h_1, \ldots, h_m)$, if $\mathcal{M}(p, h_1, \ldots, h_m) = \mathtt{true}$, namely it is a valid zero test and the answer is indeed zero, add $h_1, \ldots, h_m$ to $\widetilde{H}_{X_i,Y}$.
- Remove from $\widetilde{H}_{X_i,Y}$ all handles in $\widetilde{H}_{X_i}$.
- If $|\widetilde{H}_{X_i,Y}| \leq q_o^{\mathcal{Y}}(C, \lambda)$, add $\widetilde{H}_{X_i,Y}$ to $\widetilde{H}_Y$. Otherwise discard $\widetilde{H}_{X_i,Y}$.

Store the resulting $\widetilde{H}_Y$.

- **Output:**

$$\widetilde{C}^\star = (\widetilde{Z}, \{\widetilde{H}_X\}_{X \in \mathcal{X}_n}, \{\widetilde{H}_Y\}_{Y \in \mathcal{Y}_n}), \text{ where } \widetilde{Z} = (\{\widetilde{C}_X\}_{X \in \mathcal{X}_n}, \{\widetilde{C}_Y\}_{Y \in \mathcal{Y}_n}) \ .$$

The Evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$: Given an obfuscation $\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}\right)$, input $(x, y) \in \boldsymbol{X}_n \times \boldsymbol{Y}_n$, $\mathsf{xiO}^\star.\mathsf{Eval}^{\mathcal{M}}(\widetilde{C}^\star, (x, y))$does the following:

- Let $(X, Y) \in \mathcal{X}_n \times \mathcal{Y}_n$ be the (unique) sets such that $(x, y) \in X \times Y$.
- Emulate $\mathsf{xiO.Eval}^{(\cdot)}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$.
- Whenever $\mathsf{xiO.Eval}$ makes a zero-test query $(p, h_1, \ldots, h_m)$:
  - If for some i, $h_i \notin \widetilde{H}_X \cup \widetilde{H}_Y$, answer `false` .
  - Forward any other zero test to the oracle $\mathcal{M}$ and return its answer.

4.1.2 Analysis of Succinctness

Proposition 4.1. *The sizes of the explicit handle sets are bounded as follows*

$$q_h^{\mathcal{X}} := \max_{X \in \mathcal{X}_n} |\widetilde{H}_X| \leq O\left(q_o^{\mathcal{X}}\right) \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right) \ ,$$
$$q_h^{\mathcal{Y}} := \max_{Y \in \mathcal{Y}_n} |\widetilde{H}_Y| \leq O\left(q_o^{\mathcal{Y}}\right) \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right) \ .$$

Proof. For every $X \in \mathcal{X}_n$, $\widetilde{H}_X$ is constructed in $K_{\mathcal{X}}$ iterations when learning heavy handles for $\mathcal{X}$. In each such iteration all handles added are from the set of handles $H_X \cup H_{Y_i}$, for some $Y_i \in \mathcal{Y}_n$, generated during obfuscation and stored in the lists L_X, L_{Y_i}. Thus,

$$\begin{aligned} |\widetilde{H}_X| &\leq (|H_X| + |H_Y|) \cdot K_{\mathcal{X}} \leq \\ &\left(q_o^{\mathcal{X}} + q_o^{\mathcal{Y}}\right) \cdot \min\left(400 q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log\left(400 q_o^{\mathcal{X}}\right)\right) \leq \\ &O\left(q_o^{\mathcal{X}}\right) \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right) \ , \end{aligned}$$

where in the last inequality, we rely on our assumption that $q_o^{\mathcal{X}} \geq q_o^{\mathcal{Y}}$.

For every $Y \in \mathcal{Y}_n$, $\widetilde{H}_Y$ is constructed in $K_\mathcal{Y}$ iterations when learning heavy handles for $\mathcal{Y}$. In each such iteration, we add a set of handles of size at most $q_o^\mathcal{Y}$. Thus,

$$
\begin{aligned}
|\widetilde{H}_Y| \leq & q_o^\mathcal{Y} \cdot K_\mathcal{Y} \leq \\
& q_o^\mathcal{Y} \cdot \min\left(200 q_o^\mathcal{Y}, |\mathcal{X}_n| \cdot \log\left(200 q_o^\mathcal{Y}\right)\right) \leq \\
& O\left(q_o^\mathcal{Y}\right) \cdot \min\left(q_o^\mathcal{Y}, |\mathcal{X}_n| \cdot \log q_o^\mathcal{Y}\right) \quad .
\end{aligned}
$$

□

4.1.3 Analysis of Approximate Correctness

Proposition 4.2. *The new scheme* $\mathsf{xiO}^\star$ *satisfies the strutural requirement of explicit handles in product and is approximately correct.*

Proof. Throughout, we keep calling a zero test $(p, h_1, \ldots, h_m)$ made to oracle $\mathcal{M}$ a *true zero-test* if $\mathcal{M}(p, h_1, \ldots, h_m) = \texttt{true}$. Recall that satisfying the structural requirement of explicit handles in product form means that true zero-test queries $(p, h_1, \ldots, h_m)$ made by the evaluation algorithm $\mathsf{xiO}^\star.\mathsf{Eval}^\mathcal{M}\left(\widetilde{C}, (x, y)\right)$ should be such that for all $j \in [m]$, $h_j \in \widetilde{H}_X \cup \widetilde{H}_Y$, where $(X, Y) \in \mathcal{X}_n \times \mathcal{Y}_n$ are the (unique) sets such that $(x, y) \in X \times Y$. Indeed, by definition the evaluation algorithm $\mathsf{xiO}^\star.\mathsf{Eval}^\mathcal{M}$ emulates $\mathsf{xiO}.\mathsf{Eval}^{(\cdot)}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ and forwards to the oracle $\mathcal{M}$ only those queries that satisfy this condition.

We now turn to prove approximate correctness. In what follows, for any $X \in \mathcal{X}_n$, we denote by H_X the handles generated during obfuscation when $\mathsf{xiO}^\star.\mathsf{Obf}$ emulates $\mathsf{xiO}.\mathsf{Obf}^\mathcal{M}_\mathcal{X}(C, X, 1^\lambda)$. For $Y \in \mathcal{Y}_n$, H_Y is defined similarly. We first note that, since the original scheme $\mathsf{xiO}.\mathsf{Obf}$ is correct, for any $(x, y) \in X \times Y$, except with negligible probability, the evaluation algorithm $\mathsf{xiO}^\star.\mathsf{Eval}^\mathcal{M}(\widetilde{C}^\star, (x, y))$ may only err when the emulated $\mathsf{xiO}.\mathsf{Eval}^{(\cdot)}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ makes a true zero test that includes some handle $h_j \notin \widetilde{H}_X \cup \widetilde{H}_Y$. In this case, $\mathsf{xiO}^\star.\mathsf{Eval}$ answers `false` , possibly unlike $\mathcal{M}$. When such an error actually occurs, one of the following two must hold:

- The handle h_j was generated during obfuscation when generating $\widetilde{C}_X, \widetilde{C}_Y$, but was not added to $\widetilde{H}_X \cup \widetilde{H}_Y$ during the learning of heavy handles:

$$
h_j \in (H_X \cup H_Y) \setminus \left(\widetilde{H}_X \cup \widetilde{H}_Y\right) \quad .
$$

- The handle h_j was generated during obfuscation of some other components $\widetilde{C}_{X'}, \widetilde{C}_{Y'}$ and was not added to $\widetilde{H}_X \cup \widetilde{H}_Y$ during the learning phase:

$$
h_j \in (H_{X'} \cup H_{Y'}) \setminus \left(\widetilde{H}_X \cup \widetilde{H}_Y \cup H_X \cup H_Y\right) \quad ,
$$

  for some $X' \in \mathcal{X} \setminus \{X\}, Y' \in \mathcal{Y} \setminus \{Y\}$.

We first argue that the second case above occurs only with negligible probability.

Claim 4.1. *The probability that after executing the obfuscator* $\mathsf{xiO}.\mathsf{Obf}^\mathcal{M}(C, 1^\lambda)$*, there exists* $X, X' \in \mathcal{X}_n, Y, Y' \in \mathcal{Y}_n$*, such that* $X \neq X', Y \neq Y'$*, but* $\mathsf{xiO}.\mathsf{Eval}^\mathcal{M}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ *performs a zero test that includes a handle* $h_j \in (H_{X'} \cup H_{Y'}) \setminus (H_X \cup H_Y)$ *is at most* $2^{-\Omega(\lambda)}$.

Proof. Indeed, for any X, Y, X', Y' as above, any handle $h \in H_{X'} \cup H_{Y'}$ created when generating the obfuscations $\widetilde{C}_{X'}, \widetilde{C}_{Y'}$ is such that $h = (r, \ell)$, where $r \in \{0,1\}^{\log|\mathbb{F}|}$ is a random string sampled independently of $\widetilde{C}_X, \widetilde{C}_Y$ and thus also independently of h_j. The probability that $h_j = h$ is thus at most $1/|\mathbb{F}|$. Overall, the probability that this occurs can be union bounded by

$$\frac{(|\mathcal{X}_n| \cdot |\mathcal{Y}_n|)^2 \cdot \left(q_o^{\mathcal{X}} + q_o^{\mathcal{Y}}\right)}{|\mathbb{F}|} = \frac{\text{poly}(2^n, \lambda)}{2^{\Omega(\lambda)}} = \frac{\text{poly}(2^{O(\log\lambda)}, \lambda)}{2^{\Omega(\lambda)}} = 2^{-\Omega(\lambda)} \ .$$

□

From hereon, we focus on bounding the probability of the first event, which we shall denote by $\mathsf{Bad}(X, Y)$. That is $\mathsf{Bad}(X, Y)$ is the event that $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ makes a true zero-test query $(p, h_1, \ldots, h_m)$ where some $h_j \in (H_X \cup H_Y) \setminus \left(\widetilde{H}_X \cup \widetilde{H}_Y\right)$. (We sometimes say that such a handle h_j was *not covered* when learning heavy handles.) We note that since all sets $\mathcal{X}_n, \mathcal{Y}_n$ consist of equal-size, disjoint sets that partition $\boldsymbol{X}_n, \boldsymbol{Y}_n$, it holds that

$$\Pr_{\substack{\mathsf{xiO}^\star.\mathsf{Obf} \\ (X,Y)\leftarrow\mathcal{X}_n\times\mathcal{Y}_n}}[\mathsf{Bad}(X,Y)] = \Pr_{\substack{\mathsf{xiO}^\star.\mathsf{Obf} \\ (x,y)\leftarrow\boldsymbol{X}_n\times\boldsymbol{Y}_n}}[\mathsf{Bad}(X,Y) \mid (x,y) \in X \times Y] \ .$$

So to deduce approximate correctness it suffices to bound the probability on the left. We will show that

$$\Pr_{\substack{\mathsf{xiO}^\star.\mathsf{Obf} \\ (X,Y)\leftarrow\mathcal{X}_n\times\mathcal{Y}_n}}[\mathsf{Bad}(X,Y)] \leq 1/100 \ .$$

We further define $\mathsf{Bad}_{\mathcal{X}}(X, Y)$ to be the event that some $h_j \in H_X \setminus \widetilde{H}_X$ (namely, h_j was not covered when learning heavy handles for $\mathcal{X}$). We similarly define $\mathsf{Bad}_{\mathcal{Y}}(X, Y)$ to be the event that some $h_j \in H_Y \setminus \widetilde{H}_Y$. It suffices to bound the probability of each of the two since

$$\mathsf{Bad}(X,Y) \subseteq \mathsf{Bad}_{\mathcal{X}}(X,Y) \cup \mathsf{Bad}_{\mathcal{Y}}(X,Y) \ .$$

Bounding $\mathsf{Bad}_{\mathcal{X}}(X, Y)$. We start by bounding the probability that $\mathsf{Bad}_{\mathcal{X}}(X, Y)$ occurs. We, in fact, bound it for any fixed X, which directly the same bound for a random X.

Fix any $X \in \mathcal{X}$ and fix all sets $\{H_X\}_{X\in\mathcal{X}_n}$, $\{H_Y\}_{Y\in\mathcal{Y}_n}$ of handles and $\left\{\widetilde{C}_X\right\}_{X\in\mathcal{X}_n}, \left\{\widetilde{C}_Y\right\}_{Y\in\mathcal{Y}_n}$, generated in the obfuscation phase, when emulating $\mathsf{xiO.Obf}^{\mathcal{M}}(C, X, 1^\lambda), \mathsf{xiO.Obf}^{\mathcal{M}}(C, Y, 1^\lambda)$. Also, fix any handle $h_j \in H_X$ and consider the event $\mathsf{Bad}_{\mathcal{X}}(X, Y, h_j)$ that $\mathsf{Bad}_{\mathcal{X}}(X, Y)$ occurs with respect to the handle h_j; namely, $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ makes a true zero-test query including a handle $h_j \in H_X \setminus \widetilde{H}_X$.

Note that the event $\mathsf{Bad}_{\mathcal{X}}(X, Y, h_j)$ occurs only if during the learning of heavy handles for $\mathcal{X}$, in all iterations $i \in [K_{\mathcal{X}}]$, when sampling $Y_i \leftarrow \mathcal{Y}_n$, and emulating $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_{Y_i})$, there is no true zero-test that includes the handle h_j, whereas in the actual (post obfuscation) evaluation h_j is included in some true zero-test. Thus, denoting by α the probability, over $Y_i \leftarrow \mathcal{Y}_n$, that h_j is part of a true zero test in any single execution, we have that

$$\Pr[\mathsf{Bad}_{\mathcal{X}}(X,Y,h_j)] \leq (1-\alpha)^{K_{\mathcal{X}}} \cdot \alpha \leq \min\left(\frac{1}{K_{\mathcal{X}}}, 2^{-\frac{K_{\mathcal{X}}}{|\mathcal{Y}_n|}}\right) \leq \frac{1}{400 q_o^{\mathcal{X}}} \ ,$$

where we use Fact 2.1 and our choice of $K_{\mathcal{X}}$.

Overall, by a union bound over all $h_j \in H_X$,

$$\Pr[\mathsf{Bad}_{\mathcal{X}}(X,Y)] \le |H_X| \cdot \frac{1}{400 q_o^{\mathcal{X}}} \le \frac{1}{400} \ .$$

Bounding $\mathsf{Bad}_{\mathcal{Y}}(X,Y)$**.** To bound the probability that $\mathsf{Bad}_{\mathcal{Y}}(X,Y)$ occurs, we will bound the event

$$\Delta(X,Y) := \mathsf{Bad}_{\mathcal{Y}}(X,Y) \cap \overline{\mathsf{Bad}}_{\mathcal{X}}(X,Y)$$

that, during evaluation, $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ performs a true zero-test query that includes $h_j \in H_Y \setminus \widetilde{H}_Y$, and in addition, for all handles $h \in H_X$ included in any true zero-test, it is the case that $h \in \widetilde{H}_X$, namely they were covered during the learning of heavy handles for $\mathcal{X}$. Then, we will use the fact that

$$\Pr[\mathsf{Bad}_{\mathcal{Y}}(X,Y)] \le \Delta(X,Y) + \Pr[\mathsf{Bad}_{\mathcal{X}}(X,Y)] \ \ ,$$

and our already established bound on $\mathsf{Bad}_{\mathcal{X}}(X,Y)$.

We, in fact, bound $\Delta(X,Y)$ for any fixed Y, which directly implies the same bound for a random Y. Fix any $Y \in \mathcal{Y}$ and fix all sets $\{H_X\}_{X \in \mathcal{X}_n}, \{H_Y\}_{Y \in \mathcal{Y}_n}$ of handles and $\left\{\widetilde{C}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{C}_Y\right\}_{Y \in \mathcal{Y}_n}$, generated in the obfuscation phase, when emulating $\mathsf{xiO.Obf}^{\mathcal{M}}(C, X, 1^\lambda), \mathsf{xiO.Obf}^{\mathcal{M}}(C, Y, 1^\lambda)$. Fix also all the sets $\left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}$ of handles created during obfuscation, when learning the heavy handles for $\mathcal{X}$. Finally, fix any handle $h_j \in H_Y$ and consider the event $\Delta(X,Y,h_j)$ that $\Delta(X,Y)$ occurs with respect to the handle h_j; namely, $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X, \widetilde{C}_Y)$ makes a true zero-test query including a handle $h_j \in H_Y \setminus \widetilde{H}_Y$.

Note that the event $\Delta(X,Y,h_j)$ occurs only if during the learning of heavy handles for $\mathcal{Y}$, in all iterations $i \in [K_{\mathcal{Y}}]$, when sampling $X_i \leftarrow \mathcal{X}_n$, and emulating $\mathsf{xiO.Eval}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_{X_i}, \widetilde{C}_Y)$, it is the case that either:

- no true zero-test includes the handle h_j, or
- some true zero-test includes a handle $h \in H_X \setminus \widetilde{H}_X$.

whereas in the actual (post obfuscation) evaluation the exact opposite occurs: h_j is included in some true zero test and there is no handle $h \in H_X \setminus \widetilde{H}_X$ included in any true zero-test. Indeed, note that if in some execution $i \in [K_{\mathcal{Y}}]$, h_j is included in some true zero test and no true zero-test includes $h \in H_{X_i} \setminus \widetilde{H}_{X_i}$, then in this iteration,

$$|\widetilde{H}_{X_i,Y}| \le |H_Y| \le q_o^{\mathcal{Y}} \ \ ,$$

and the learning procedure would have added h_j to $\widetilde{H}_Y$.

Thus, denoting by β the probability, over $X_i \leftarrow \mathcal{X}_n$, that in any single evaluation, h_j is part of a true zero test and no true zero-test includes handles $h \in H_{X_i} \setminus \widetilde{H}_{X_i}$, we have that

$$\Pr[\Delta(X,Y,h_j)] \le (1-\beta)^{K_{\mathcal{Y}}} \cdot \beta \le \min\left(\frac{1}{K_{\mathcal{Y}}}, 2^{-\frac{K_{\mathcal{Y}}}{|\mathcal{X}_n|}}\right) \le \frac{1}{200 q_o^{\mathcal{Y}}} \ \ ,$$

where we use Fact 2.1 and our choice of $K_{\mathcal{Y}}$.

Overall, by a union bound over all $h_j \in H_Y$,

$$\Pr[\Delta(X,Y)] \le |H_Y| \cdot \frac{1}{200 q_o^{\mathcal{Y}}} \le \frac{1}{200} \ .$$

Overall, we have

$$\begin{aligned}\Pr\left[\mathsf{Bad}(X,Y)\right] \leq &\Pr\left[\mathsf{Bad}_{\mathcal{X}}(X,Y)\right] + \Pr\left[\mathsf{Bad}_{\mathcal{Y}}(X,Y)\right] \leq \\ &\Pr\left[\mathsf{Bad}_{\mathcal{X}}(X,Y)\right] + \Delta(X,Y) + \Pr\left[\mathsf{Bad}_{\mathcal{X}}(X,Y)\right] \leq \\ &1/100 \ .\end{aligned}$$

□

4.1.4 Analysis of Security

Proposition 4.3. *The new scheme* $\mathsf{xiO}^\star$ *is as secure as the original scheme* xiO.

Proof sketch. The new obfuscation $\widetilde{C}^\star$ consists exactly of an obfuscation $\widetilde{C}$ under the original scheme, plus the handles $\left\{\widetilde{H}_X\right\}_{X\in\mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y\in\mathcal{Y}_n}$, which were generated in the learning phase. The leraning phase, consists of running the evaluation algorithm of the original scheme repeatedly for a polynomial number of times, with a slight difference — when emulating an evaluation $\mathsf{xiO.Eval}^{\mathcal{M}}_{\mathcal{X},\mathcal{Y}}(\widetilde{C}_X,\widetilde{C}_Y)$ oracle queries are answered using the specific lists (L_X, L_Y), ignoring the other stored lists $L_{X'}, L_{Y'}$, for $X \neq X', Y \neq Y'$. This may be different from the original evaluation if a zero test that includes a handle $h_j \in H_{X'} \cup H_{Y'} \setminus (H_X \cup H_Y)$ is performed. However, as we have already shown in Claim 4.1, this occurs with negligible probability $2^{-\lambda}$. Thus, any attacker against the new scheme can be statistically simulated by an attacker for the original scheme that simulates the above handles by performing the learning process by itself. □

This concludes the proof of Lemma 4.1.

4.2 Step 2: From Constant Degree to Degree Two

We show that any XIO scheme with explicit handles in product form, relative to a degree-d decomposable ideal oracle (for arbitrary $d = O(1)$), can be transformed into one relative to a degree-2 decomposable ideal oracle. The resulting degree-2 oracle is defined with respect to a validity predicate V^2 related to the validity predicate V^d of the degree-d oracle we start with.

Intuitively, this model can be seen as an extension of the standard asymmetric bilinear maps, where instead of two base groups we may have more. That is, instead of two asymmetric base-groups G_1, G_2 where $(g_1^a, g_2^b) \in G_1 \times G_2$ can be mapped to $e(g_1, g_2)^{ab}$ in the target group G_T, we possibly have a larger number of groups $G_1, \dots, G_n$ and a collection of *valid mappings* $\{e_k : G_{i_k} \times G_{j_k} \to G_T\}$, which may be a strict subset of all possible bilinear maps.

In the next section, we show another transformation that further reduces the oracle to one *with no labels at all*, analogous to the symmetric bilinear case where there is a single base group G and a single map $e : G \times G \to G_T$.

Lemma 4.2. *let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{(\cdot)}, \mathsf{xiO.Eval}^{(\cdot)})$ *be an XIO scheme, for a collection of circuit classes* $\mathcal{C}$*, defined relative to a degree-*d *decomposable ideal oracle* $\mathcal{M}^d = \{\mathcal{M}^d_{\mathbb{F}_\lambda, V_\lambda}\}$*, with explicit handles in* $(\mathcal{X},\mathcal{Y})$*-product form, for some product collection* $(\mathcal{X},\mathcal{Y})$*. Assume further that for some constant* $\gamma < 1$*,*

$$|\mathcal{X}_n| \cdot \left(q_h^{\mathcal{X}}\right)^d + |\mathcal{Y}_n| \cdot \left(q_h^{\mathcal{Y}}\right)^d \leq 2^{\gamma n} \cdot \mathrm{poly}(|C|, \lambda) \ .$$

Then xiO *can be converted to a new scheme* $\mathsf{xiO}^\star$*, also with explicit handles in* $(\mathcal{X},\mathcal{Y})$*-product form, relative to a degree-*2 *decomposable oracle* $\mathcal{M}^2$.

To prove the above lemma, we first present our new XiO scheme relative to a degree-2 decomposable oracle in Section 4.2.1; we analyze the succinctness of the new scheme in Section 4.2.2, and show that the new XiO scheme is as (approximately) correct and secure as the original XiO scheme relative to a degree-d decomposable oracle in Section 4.2.3.

4.2.1 The New XiO Scheme Relative to a Degree-2 Oracle $\mathcal{M}^2$

In what follows, let $\mathsf{xiO} = (\mathsf{xiO.Obf}^{(\cdot)}, \mathsf{xiO.Eval}^{(\cdot)})$ be an XIO scheme with explicit handles in product form, defined relative to a degree-d decomposable ideal oracle $\mathcal{M}^d = \{\mathcal{M}^d_{\mathbb{F}_\lambda, V_\lambda}\}$. We describe a new scheme $\mathsf{xiO}^\star = (\mathsf{xiO}^\star.\mathsf{Obf}^{(\cdot)}, \mathsf{xiO}^\star.\mathsf{Eval}^{(\cdot)})$ (also, with explicit handles in product form) defined relative to a degree-2 decomposable ideal oracle $\mathcal{M}^2 = \{\mathcal{M}^2_{\mathbb{F}_\lambda, V^\star_\lambda}\}$.

The Obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}$**:** Given a circuit $C \in \boldsymbol{\mathcal{C}}$ with input size n, and security parameter 1^λ, and oracle access to $\mathcal{M}^2$, $\mathsf{xiO}^\star.\mathsf{Obf}^{\mathcal{M}^2}(C, 1^\lambda)$ does as follows:

- **Emulate Obfuscation:**
  - Emulate $\mathsf{xiO.Obf}^{\mathcal{M}^d}(C, 1^\lambda)$.
  - Throughout the emulation, emulate the oracle $\mathcal{M}^d$, storing a list $L = \{(h, \xi)\}$ of encoded element-label pairs (ξ, ℓ) and corresponding handles $h = (r, \ell)$.
  - Obtain the obfuscation $(\widetilde{Z}, \{\widetilde{H}_X\}_{X \in \mathcal{X}_n}, \{\widetilde{H}_X\}_{Y \in \mathcal{Y}_n})$.
- **Encode Monomials:**
  - For each $X \in \mathcal{X}_n$:
    1. Retrieve $\widetilde{H}_X = (h_1, \ldots, h_m)$ and the corresponding field elements and labels $(\xi_1, \ell_1), \ldots, (\xi_m, \ell_m)$ from the stored list L.
    2. For every formal monomial $\Phi(v_1, \ldots, v_m) = v_{i_1} \ldots v_{i_j}$, where $j \le d$ and $i_1, \ldots, i_j \in [m]$,
       * Compute
       $$\Phi(\boldsymbol{\xi}) := \xi_{i_1} \cdot \xi_{i_2} \cdots \xi_{i_j}, \qquad \Phi(\boldsymbol{\ell}) := \left\{\ell_{i_1}, \ldots, \ell_{i_j}\right\}, \qquad \Phi(\boldsymbol{h}) := \left\{h_{i_1}, \ldots, h_{i_j}\right\} .$$
       (For simplicity of notation, we overload Φ to describe different functions when acting on field elements, labels, and handles.)
       * Request $\mathcal{M}^2$ to encode the field element and label $(\xi^\star_{X,\Phi}, \ell^\star_{X,\Phi}) := (\Phi(\boldsymbol{\xi}), \Phi(\boldsymbol{\ell}))$, and obtain a handle $h^\star_{X,\Phi}$.
    3. Store $\widetilde{H}^\star_X = \left\{(h^\star_{X,\Phi}, \Phi(\boldsymbol{h}))\right\}_\Phi$
  - For each $Y \in \mathcal{Y}_n$:
    1. Symmetrically perform the above two steps with respect to $\widetilde{H}_Y$ (instead of $\widetilde{H}_X$).
    2. Store $\widetilde{H}^\star_Y = \left\{(h^\star_{Y,\Phi}, \Phi(\boldsymbol{h}))\right\}_\Phi$.
- **Output:**
  - $\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}^\star_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}^\star_Y\right\}_{Y \in \mathcal{Y}_n}\right)$, where $\widetilde{C} := \left(\widetilde{Z}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}\right)$.

The Evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$**:** Given an obfuscation $\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}^\star_X\right\}_{X\in\mathcal{X}_n}, \left\{\widetilde{H}^\star_Y\right\}_{Y\in\mathcal{Y}_n}\right)$, input $(x,y) \in \boldsymbol{X}_n \times \boldsymbol{Y}_n$, and oracle access to $\mathcal{M}^2$, $\mathsf{xiO}^\star.\mathsf{Eval}^{\mathcal{M}^2}(\widetilde{C}^\star, (x,y))$ does the following:

- Emulate $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{M}^d}(\widetilde{C}, (x,y))$.
- Emulate any zero-test query $(p, h_1, \dots, h_m)$ it makes to $\mathcal{M}^d$ as follows:
  1. Parse $\widetilde{C} = (\widetilde{Z}, \{\widetilde{H}_X\}_{X\in\mathcal{X}_n}, \{\widetilde{H}_Y\}_{Y\in\mathcal{Y}_n})$.
  2. Let $(X,Y) \in \mathcal{X}_n \times \mathcal{Y}_n$ be the (unique) sets such that $(x,y) \in X \times Y$. Retrieve $\widetilde{H}_X, \widetilde{H}_Y$.
  3. Split $\boldsymbol{h} = (h_1, \dots, h_m)$ into two verctors of handles $\boldsymbol{h}_X \subseteq \widetilde{H}_X$ and $\boldsymbol{h}_Y \subseteq \widetilde{H}_Y$. (Such a partition always exists, by the guarantee of explicit handles in product form.)
  4. Viewing $p(\boldsymbol{h})$ as a formal polynomial in variables $\boldsymbol{h}$, factor it as
  $$p(\boldsymbol{h}) = \sum_i \gamma_i \Phi_i(\boldsymbol{h}) = \sum_i \gamma_i \Phi_{X,i}(\boldsymbol{h}_X)\Phi_{Y,i}(\boldsymbol{h}_Y) \ ,$$
  where $\gamma_i \in \mathbb{F} \setminus \{0\}$ are the coefficients, and each monomial $\Phi_i(\boldsymbol{h})$ is factored into $\Phi_{X,i}(\boldsymbol{h}_X) \cdot \Phi_{Y,i}(\boldsymbol{h}_Y)$.
  5. Translate $\{\Phi_{X,i}(\boldsymbol{h}_X), \Phi_{Y,i}(\boldsymbol{h}_Y)\}_i$ into handles $\{h^\star_{X,i}, h^\star_{Y,i}\}_i$ by locating $(h^\star_{X,i}, \Phi_{X,i}(\boldsymbol{h}_X)) \in \widetilde{H}^\star_X$ and $(h^\star_{Y,i}, \Phi_{Y,i}(\boldsymbol{h}_Y)) \in \widetilde{H}^\star_Y$.
  6. Consider the degree-2 formal polynomial:
  $$p^\star(\boldsymbol{h}^\star) = \sum_i \gamma_i h^\star_{X,i} h^\star_{Y,i} \ .$$
  7. Make the zero-test $(p^\star, \boldsymbol{h}^\star)$ to the oracle $\mathcal{M}^2$ and return the result.

Labels and Validity Predicate V^2 of Oracle $\mathcal{M}^2$. Note that labels with respect to $\mathcal{M}^2$ are subsets of the label set of $\mathcal{M}$. Let V^d be the decomposable validity predicate associated with $\mathcal{M}^d$. We define a new validity predicate of degree 2, which is also decomposable. For this purpose, we need to define V^2 for labels corresponding to bilinear monomials given by a multi-set of cardinality at most 2, namely of the form $\{\ell_1^\star, \ell_2^\star\}$ or $\{\ell^\star\}$. For multi-sets L with cardinality larger than 2, we define $V^2(L) = \texttt{false}$, capturing that this is a degree 2-predicate.

For a multi-set $\{\ell_1^\star, \ell_2^\star\}$, the validity predicate $V^2(\{\ell_1^\star, \ell_2^\star\})$ is computed as follows:

- Parse $\ell_1^\star$ and $\ell_2^\star$ as as two multi-sets $\{\ell_{1,1}, \dots, \ell_{1,k_1}\}$, $\{\ell_{2,1}, \dots, \ell_{2,k_2}\}$.
- Apply the original predicate to the union (with multiplicity) multi-set:

$$V^2(\{\ell_1^\star, \ell_2^\star\}) := V^d(\ell_1^\star \uplus \ell_2^\star) = V^d(\{\ell_{1,1}, \dots, \ell_{1,k_1}\} \uplus \{\ell_{2,1}, \dots, \ell_{2,k_2}\}) \ .$$

For a multi-set $\{\ell^\star\}$ of cardinality one, we define:

$$V^2(\{\ell^\star\}) := V^d(\ell^\star) \ .$$

Recall that the fact that V^d is decomposable means that there exist a projection function Π^d and predicate V^d_Π, such that, for every two multi-sets $\ell_1^\star, \ell_2^\star$,

$$V^d(\ell_1^\star \uplus \ell_2^\star) = V^d_\Pi(\Pi^d(\ell_1^\star), \Pi^d(\ell_2^\star)) \ .$$

We show that V^2 is also decomposable. For this, we define its corresponding projection function Π^2 and predicate V^2_Π, and show that on input two multi-sets $A = \{\ell_i^\star\}_i$ and $B = \{\ell_j^\star\}_j$,

$$V^2(A \uplus B) = V^2_\Pi(\Pi^2(A), \Pi^2(B)) \ .$$

The projection function Π^2 on input a multi-set A computes:

$$\Pi^2(A) = \Big(|A|, \Pi^d(\uplus_{\ell^\star \in A} \ell^\star)\Big) \ .$$

The predicate V^2_Π on input two multi-sets A, B outputs `false` if $|A| + |B| > 2$. Otherwise, if A, B contain exactly two labels $\ell_1^\star, \ell_2^\star$, the predicate computes:

$$\begin{aligned} V^2_\Pi(\Pi^2(A), \Pi^2(B)) &= V^d(\Pi^d(\uplus_{\ell^\star \in A} \ell^\star), \Pi^d(\uplus_{\ell^\star \in B} \ell^\star)) \\ &= V^d((\uplus_{\ell^\star \in A} \ell^\star) \uplus (\uplus_{\ell^\star \in B} \ell^\star)) \\ &= V^d(\ell_1^\star \uplus \ell_2^\star) \\ &= V^2(A \uplus B) \ . \end{aligned}$$

Therefore V^2 is decomposable. Moreover, it is easy to see that the arity of V^2 is exactly that of V^d, which is bounded by a fixed polynomial.

4.2.2 Analysis of Succinctness

Proposition 4.4. *The size of the new obfuscation $|\widetilde{C}^\star|$ is bounded by that of the original obfuscation $|\widetilde{C}|$*

$$\textit{plus} \ \left(|\mathcal{X}_n| \cdot \left(q_h^{\mathcal{X}}\right)^d + |\mathcal{Y}_n| \cdot \left(q_h^{\mathcal{Y}}\right)^d\right) \cdot O(\lambda) \ .$$

Proof. Recall that

$$\begin{aligned} &\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}^\star_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}^\star_Y\right\}_{Y \in \mathcal{Y}_n}\right) \text{ where for any } X \in \mathcal{X}_n, Y \in \mathcal{Y}_n \\ &\widetilde{H}^\star_X = \left\{(h_{X,\Phi}, \Phi(\boldsymbol{h})) \ \middle|\ \Phi(\boldsymbol{h}) \text{ is a degree-}d \text{ monomial in } \boldsymbol{h} = \widetilde{H}_X\right\} \ , \\ &\widetilde{H}^\star_Y = \left\{(h_{Y,\Phi}, \Phi(\boldsymbol{h})) \ \middle|\ \Phi(\boldsymbol{h}) \text{ is a degree-}d \text{ monomial in } \boldsymbol{h} = \widetilde{H}_Y\right\} \ . \end{aligned}$$

The bound follows from the fact that the number of degree-d monomials in $\widetilde{H}_X$ (respectively, $\widetilde{H}_Y$) is at most $(|\widetilde{H}_X| + 1)^d \leq (q_h^{\mathcal{X}} + 1)^d$ (respectively, $(q_h^{\mathcal{Y}} + 1)^d$) and the fact that each handle is a string of size $O(\lambda)$. □

4.2.3 Analysis of Correctness and Security

Proposition 4.5. *The new scheme* $\mathsf{xiO}^\star$ *is as correct and as secure as the original scheme* xiO. *In particular, if* xiO *is an approximately-correct XIO so is* $\mathsf{xiO}^\star$.

Proof. To prove this, we show a PPT simulator $\mathcal{S}$ that given access to a degree-d oracle $\mathcal{M}^d$ and an obfuscation $\widetilde{C} \leftarrow \mathsf{xiO.Obf}^{\mathcal{M}^d}(C, 1^\lambda)$, for an arbitrary circuit $C \in \boldsymbol{\mathcal{C}}$, can perfectly simulate the view of any polynomial-size adversary $\mathcal{A}$ that gets oracle access to the degree-2 oracle $\mathcal{M}^2$ and an obfuscation $\widetilde{C}^\star \leftarrow \mathsf{xiO}^\star.\mathsf{Obf}^{\mathcal{M}^2}(C, 1^\lambda)$:

$$C, \mathcal{S}^{\mathcal{M}^d, \mathcal{A}^{(\cdot)}}(\widetilde{C}) \equiv C, \mathcal{A}^{\mathcal{M}^2}(\widetilde{C}^\star) \ .$$

The existence of $\mathcal{S}$ indeed implies $\mathsf{xiO}^\star$ is as secure as the original scheme xiO. To show correctness, we will consider an adversary that simply performs evaluation in the new scheme $\mathsf{xiO}^\star$, and argue that the corresponding simulator emulates evaluation in the original scheme xiO. (See details below.)

The simulator $\mathcal{S}$ given $\widetilde{C} := \left(\widetilde{Z}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}\right)$, performs the following:

1. Emulates the encoding step of the obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}^{\mathcal{M}^2}$:
   - For each $X \in \mathcal{X}_n$, retrieves $\widetilde{H}_X = (h_1, \ldots, h_m)$ and for every formal monomial $\Phi(v_1, \ldots, v_m) = v_{i_1} \ldots v_{i_j}$ of degree at most d, computes $\Phi(\boldsymbol{h}) := \left\{h_{i_1}, \ldots, h_{i_j}\right\}$ and $\Phi(\boldsymbol{\ell}) := \left\{\ell_{i_1}, \ldots, \ell_{i_j}\right\}$, where ℓ_i is the label associated with $h_i = (r_i, \ell_i)$.
   - Emulates the handle $h^\star_{X,\Phi} = (r^\star_{X,\Phi}, \Phi(\boldsymbol{\ell}))$, where $r^\star_{X,\Phi}$ is sampled at random.
   - Stores $\widetilde{H}^\star_X = \left\{(h^\star_{X,\Phi}, \Phi(\boldsymbol{h}))\right\}_\Phi$.
   - Does the same for each $Y \in \mathcal{Y}_n$ and stores $\widetilde{H}^\star_Y$.
2. Lets $\widetilde{C}^\star := \left(\widetilde{C}, \left\{\widetilde{H}^\star_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}^\star_Y\right\}_{Y \in \mathcal{Y}_n}\right)$ and starts emulating $\mathcal{A}(\widetilde{C}^\star)$.
3. Whenever $\mathcal{A}$ makes a zero-test query $(p^\star, \boldsymbol{h}^\star)$ meant for $\mathcal{M}^2$, the simulator $\mathcal{S}$
   - Writes $p^\star$ as a formal polynomial in $\boldsymbol{h}^\star$:
     $$p^\star(\boldsymbol{h}^\star) = \sum_{i,j} \gamma_{i,j} h^\star_i h^\star_j \ .$$
     If $p^\star$ is of degree higer than 2 (and thus cannot be written as above), returns `false` .
   - Verifies that for any coordinate $h^\star_i$ of $\boldsymbol{h}^\star$, there is some monomial Φ_i, such that
     $$(h^\star_i, \Phi_i(\boldsymbol{h})) \in \left\{\widetilde{H}^\star_X\right\}_{X \in \mathcal{X}_n} \bigcup \left\{\widetilde{H}^\star_Y\right\}_{Y \in \mathcal{Y}_n} \ ,$$
     where $\boldsymbol{h} := \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n} \bigcup \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}$. Otherwise returns `false` .
   - Constructs a corresponding polynomial p in $\boldsymbol{h}$:
     $$p(\boldsymbol{h}) = \sum_{i,j} \gamma_{i,j} \Phi_i(\boldsymbol{h}) \Phi_j(\boldsymbol{h}) \ .$$
   - Makes the zero-test $(p, \boldsymbol{h})$ to $\mathcal{M}^d$ and returns the result.

We first note that the obfuscation $\widetilde{C}^{\star}$ generated by $\mathcal{S}$ perfectly emulates an obfuscation by $\mathsf{xiO}^{\star}.\mathsf{Obf}^{\mathcal{M}^2}(C, 1^{\lambda})$. Next, we observe that any zero-test $(p^{\star}, \boldsymbol{h}^{\star})$ is answered exactly as it would have been answered by $\mathcal{M}^2$.

In the scheme $\mathsf{xiO}^{\star}$, every zero-test query $(p^{\star}, \boldsymbol{h}^{\star})$ to $\mathcal{M}^2$ (over valid handles) can be translated to a zero-test $(p, \boldsymbol{h})$ for $\mathcal{M}^d$, such that if $\boldsymbol{h}^{\star}$ corresponds to $(\boldsymbol{\xi}^{\star}, \boldsymbol{\ell}^{\star})$ in $\mathsf{xiO}^{\star}$, then $\boldsymbol{h}$ corresponds to $(\boldsymbol{\xi}, \boldsymbol{\ell})$ in xiO where

$$V^2(p^{\star}, \boldsymbol{\ell}^{\star}) = V^d(p, \boldsymbol{\ell}) \ ,$$
$$p^{\star}(\boldsymbol{\xi}^{\star}) = \sum_{i,j} \gamma_{i,j} \xi_i^{\star} \xi_j^{\star} = \sum_{i,j} \gamma_{i,j} \Phi_i(\boldsymbol{\xi}) \Phi_j(\boldsymbol{\xi}) = p(\boldsymbol{\xi}) \ ; \ .$$

namely, the first is valid if and only if the latter is, and they evaluate to the same result (in the field $\mathbb{F}$). The constructed simulator $\mathcal{S}$ translates every $(p^{\star}, \boldsymbol{h}^{\star})$ to $(p, \boldsymbol{h})$ in exactly the same way, and thus perfect emulation follows.

The above simulator directly implies that if the original scheme guarantees indistinguishability so does the new scheme. We now show that it implies that the new scheme is as correct as the original one. Consider, the adversary $\mathcal{A}^{\mathcal{M}^2}(\widetilde{C}^{\star})$ that simply samples a random input $u \leftarrow \{0,1\}^n$, runs the evaluation algorithm $\mathsf{xiO}^{\star}.\mathsf{Eval}^{\mathcal{M}^2}(\widetilde{C}^{\star}, u)$, obtains the result $v^{\star}$, and outputs $(u, v^{\star})$. Observe that for this adverasry the simulator $\mathcal{S}^{\mathcal{M}^d, \mathcal{A}}(\widetilde{C})$, exactly emulates $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{M}^d}(\widetilde{C}, z)$, obtains the result v, and outputs (u, v). By the simulation guarantee (u, v) and $(u, v^{\star})$ are identically distributed, and thus the new scheme has the same (approximate) correctness as the original scheme. □

4.3 Step 3: Asymmetric Oracles to Symmetric Oracles

We show in Section 4.3.1 a transformation that reduces the oracle $\mathcal{M}^2$ (that has a complex label structure) to a symmetric bilinear oracle $\mathcal{B}^2$ (Definition 4.3). This model is analogous to the symmetric bilinear pairing groups where there is a single base group G with a bilinear map $e : G \times G \to G_T$. The transformation will incur a certain blowup depending on the arity of the oracle $\mathcal{M}^2$, which is a bounded polynomial. We then use this transformation to convert any XiO scheme relative to $\mathcal{M}^2$ to an XiO scheme relative to $\mathcal{B}^2$ in Section 4.3.2.

4.3.1 Reducing Oracle $\mathcal{M}^2$ to Oracle $\mathcal{B}^2$

The transformation consists of a recoding process $\mathcal{E}$ that takes a secret key K, and an arbitrary encoding query of the form (ξ, ℓ) to $\mathcal{M}^2$, and transforms it into a set of new encoding queries $(\xi_1^{\star}, \ell_{\mathcal{B}}), \dots, (\xi_k^{\star}, \ell_{\mathcal{B}})$ which it gives $\mathcal{B}^2$ (all with respect to the unique label $\ell_{\mathcal{B}}$). $\mathcal{E}$ then outputs a handle $\boldsymbol{h}$ representing (ξ, ℓ) consisting of a list of handles $\boldsymbol{h} = (h_1^{\star}, \dots, h_k^{\star})$ generated by $\mathcal{B}^2$ for $\xi_1^{\star}, \dots, \xi_k^{\star}$.

The encoder $\mathcal{E}$ is associated with a (public) decoder $\mathcal{D}$. The decoder $\mathcal{D}$ is given as input a zero-test query $(p, \boldsymbol{h}_1, \dots, \boldsymbol{h}_m)$ for $\mathcal{M}^2$ to be evaluated over underlying field elements $\boldsymbol{\xi} = (\xi_1, \dots, \xi_m)$, and now represented by $\boldsymbol{\xi}^{\star} = (\xi_{1,1}^{\star}, \dots, \xi_{1,k}^{\star}, \dots, \xi_{m,1}^{\star}, \dots, \xi_{m,k}^{\star})$ encoded in $\mathcal{B}^2$ with handles $\boldsymbol{h}^{\star} = (h_{1,1}^{\star}, \dots, h_{1,k}^{\star}, \dots, h_{m,1}^{\star}, \dots, h_{m,k}^{\star})$. The decoder then translates it into a new zero-test query $(p^{\star}, \boldsymbol{h}^{\star})$ and submits it to $\mathcal{B}^2$, with the guarantee that if the zero test is valid with respect to the validity predicate V associated with $\mathcal{M}^d$, then $p(\boldsymbol{\xi}) = p^{\star}(\boldsymbol{\xi}^{\star})$, and otherwise, $p^{\star}(\boldsymbol{\xi}^{\star})$ evaluates to non-zero with overwhelming probability.

We next turn to a more formal description of the transformation. In what follows, let V be an arbitrary degree-2 decomposable validity predicate, defined over multi-sets of labels of cardinality at most two taken from a label set $\mathbb{L}$, and associated with projection function Π and predicate V_Π with bounded arity $\mathsf{Arity}(V_\Pi) \leq \mathrm{poly}(\lambda)$.

Secret Encoding Key. The secret key K consists of random invertible field elements $\eta_\ell, \varphi_\ell \leftarrow \mathbb{F} \setminus \{0\}$ for each label $\ell \in \mathbb{L}$, and random invertible field elements $\alpha_\pi, \beta_\pi, \gamma_\pi, \delta_\pi \leftarrow \mathbb{F} \setminus \{0\}$ for every π in the corresponding set of projections $\Gamma = \{\Pi(\{\ell\}) \ : \ \ell \in \ \mathbb{L}\}$.

Remark 4.3 (Lazy Secret-Key Sampling). Note that the total number of labels and their projection could be superpolynomial, making the secret key superpolynomial in length. To deal with such cases, the recoder will use lazy sampling to sample the above random invertible elements only when needed and keeps a record of all sampled elements. As we argue below, the total number of random invertible elements to be sampled is polynomial in the number of tuples (ξ, ℓ) to be recoded. For simplicity of exposition, we describe the procedure with respect to a key consisting of all possible random invertible elements.

Recoding. Given the secret key K and $(\xi, \ell) \in \mathbb{F} \times \mathbb{L}$, the encoder $\mathcal{E}^{\mathcal{B}^2}((\xi, \ell), K)$ does the following:

- Samples two secret shares ξ_L, ξ_R at random from $\mathbb{F}$ subject to $\xi_L + \xi_R = \xi$.
- Let $\pi = \Pi(\{\ell\})$ be the projection of $\{\ell\}$. Generates the field elements:

$$\boldsymbol{\xi}^\star_\circ := \left(\xi^\star_{\circ,\alpha,L} = \alpha_\pi \cdot \xi_L, \quad \xi^\star_{\circ,\beta,R} = \beta_\pi \cdot \xi_R, \quad \xi^\star_{\circ,\gamma,L} = \gamma_\pi \cdot \xi_L, \quad \xi^\star_{\circ,\delta,R} = \delta_\pi \cdot \xi_R\right) \ .$$

- Let $\mathsf{match}(\pi) = \{\pi' \ : \ V_\Pi(\pi, \pi') = \mathtt{true}\}$ be the set of projections that evaluates to $\mathtt{true}$ with π. (For every $\pi' \in \mathsf{match}(\pi)$, and every ℓ', such that, $\pi' = \Pi(\{\ell'\})$, it holds that $V(\{\ell, \ell'\}) = \mathtt{true}$.)

  For each $\pi' \in \mathsf{match}(\pi)$, generates the field elements:

$$\boldsymbol{\xi}^\star_{\pi'} := \left(\xi^\star_{\pi',\frac{1}{\alpha},L} = \frac{1}{\alpha_\pi} \cdot \xi_L, \quad \xi^\star_{\pi',\frac{1}{\beta},L} = \frac{1}{\beta_\pi} \cdot \xi_L, \quad \xi^\star_{\pi',\frac{1}{\gamma},R} = \frac{1}{\gamma_\pi} \cdot \xi_R, \quad \xi^\star_{\pi',\frac{1}{\delta},R} = \frac{1}{\delta_\pi} \cdot \xi_R\right) \ .$$

- If $V(\{\ell\}) = \mathtt{true}$, generates field elements

$$\boldsymbol{\xi}^\star_\triangle := \left(\xi^\star_{\triangle,\eta,L} = \eta_\ell \cdot \xi_L, \quad \xi^\star_{\triangle,\frac{1}{\eta}} = \frac{1}{\eta_\ell}, \quad \xi^\star_{\triangle,\varphi,R} = \varphi_\ell \cdot \xi_R, \quad \xi^\star_{\triangle,\frac{1}{\varphi}} = \frac{1}{\varphi_\ell}\right) \ ,$$

- Asks $\mathcal{B}^2$ to encode (with respect to the unique label $\ell_\mathcal{B}$) the field elements

$$\boldsymbol{\xi}^\star_\circ, (\boldsymbol{\xi}^\star_{\pi'})_{\pi' \in \mathsf{match}(\pi)}, \boldsymbol{\xi}^\star_\triangle$$

  generated above, obtaining corresssponding handles

$$\boldsymbol{h}^\star = \left(\boldsymbol{h}^\star_\circ, (\boldsymbol{h}^\star_{\pi'})_{\pi' \in \mathsf{match}(\pi)}, \boldsymbol{h}^\star_\triangle\right) \ .$$

- Outputs the handles $\boldsymbol{h}^\star$.

We argue that when V has bounded $\mathrm{poly}(\lambda)$ arity, the size of the new encoding $\boldsymbol{h}^\star$ is bounded by $\mathrm{poly}(\lambda)$. This is because, $\boldsymbol{h}^\star_\circ$ and $\boldsymbol{h}^\star_\triangle$ each consists of four encodings, while $(\boldsymbol{h}^\star_{\pi'})_{\pi' \in \mathsf{match}(\pi)}$ consists of $O(|\mathsf{match}(\pi)|) = \mathsf{Arity}(V_\Pi) \leq \mathrm{poly}(\lambda)$.

Decoding. Given a degree-2 polynomial p and handles $(\boldsymbol{h}^\star_1, \ldots, \boldsymbol{h}^\star_m)$, where $\boldsymbol{h}^\star_i = \boldsymbol{h}^\star_{i,\circ}, \left(\boldsymbol{h}^\star_{i,\pi'}\right)_{\pi' \in \mathsf{match}(\pi)}, \boldsymbol{h}^\star_{i,\triangle}$ the decoder $\mathcal{D}^{\mathcal{B}^2}(p, \boldsymbol{h}^\star_1, \ldots, \boldsymbol{h}^\star_m)$:

- Writes p as a formal polynomial

$$p(\boldsymbol{h}_1^\star, \ldots, \boldsymbol{h}_m^\star) = \sigma + \sum_k \rho_k \boldsymbol{h}_k^\star + \sum_{i \le j} \rho_{i,j} \boldsymbol{h}_i^\star \boldsymbol{h}_j^\star \enspace .$$

- If for any monomial $\boldsymbol{h}_k^\star$ in p, $V(\{\ell_k\}) = \texttt{false}$, or for any monomial $\boldsymbol{h}_i^\star \boldsymbol{h}_j^\star$, $V(\{\ell_i, \ell_j\}) = \texttt{false}$, return $\texttt{false}$. Otherwise, continue.

- Generates a new degree-2 formal polynomial

$$p^\star(\boldsymbol{h}^\star) = \sigma + \sum_k \rho_k \left(h^\star_{k,\triangle,\eta,L} h^\star_{k,\triangle,\frac{1}{\eta}} + h^\star_{k,\triangle,\varphi,R} h^\star_{k,\triangle,\frac{1}{\varphi}} \right) + \sum_{i \le j} \rho_{i,j} \left(h^\star_{i,\circ,\alpha,L} h^\star_{j,\pi_i,\frac{1}{\alpha},L} + h^\star_{i,\circ,\gamma,L} h^\star_{j,\pi_i,\frac{1}{\gamma},R} + h^\star_{i,\circ,\beta,R} h^\star_{j,\pi_i,\frac{1}{\beta},L} + h^\star_{i,\circ,\delta,R} h^\star_{j,\pi_i,\frac{1}{\delta},R} \right) \enspace ,$$

where $\pi_i = \Pi(\{\ell_i\})$.

- It submits to $\mathcal{B}^2$ the zero test $(p^\star, \boldsymbol{h}^\star)$ and returns the result.

We prove the following lemma that shows that encodings of given field elements (relative to some labels), and zero-tests with respect to the above encoder $\mathcal{E}$ and symmetric oracle $\mathcal{B}^2$, do not give more power than directly encoding and zero-testing these elements with respect to the asymmetric oracle $\mathcal{M}^2$. We show this by proving the existence of an appropriate simulator. We then use this lemma show how any XIO scheme relative to $\mathcal{M}^2$ can be converted to an XIO scheme relative to $\mathcal{B}^2$.

Lemma 4.3. *Let $\mathcal{F}(Z, 1^\lambda)$ be any process (e.g., an obfuscator) that given an input Z outputs pairs $(\xi_1, \ell_1), \ldots, (\xi_m, \ell_m) \in \mathbb{F} \times \mathbb{L}$ along with auxiliary input $\widetilde{Z}$.*
Consider the following two experiments:

1. **Ideal:** *Each (ξ_i, ℓ_i) is encoded in the (asymmetric) oracle $\mathcal{M}$, resulting in a corresponding set of handles $\widetilde{H}$.*

2. **Real:** *Each (ξ_i, ℓ_i) is encoded in the (symmetric) oracle $\mathcal{B}$, using the encoder $\mathcal{E}$, resulting in a corresponding set of handles $\widetilde{H}^\star$.*

Then, there exists a PPT simulator $\mathcal{S}$ that given access to $\mathcal{M}^2$ and $\widetilde{H}$, can statistically simulate the view of any polynomial-size adversary $\mathcal{A}$ that gets oracle access to $\mathcal{B}^2$ and $\widetilde{H}^\star$:

$$Z, \mathcal{S}^{\mathcal{M}^2, \mathcal{A}^{(\cdot)}}(\widetilde{Z}, \widetilde{H}) \approx_s Z, \mathcal{A}^{\mathcal{B}^2}(\widetilde{Z}, \widetilde{H}^\star) \enspace .$$

Proof of Lemma 4.3. We start by describing the simulator $\mathcal{S}$ and then analyze it.

The simulator $\mathcal{S}$ given $\widetilde{Z}$ and $\widetilde{H} = \{h_1, \ldots, h_m\}$, performs the following:

1. Translates each $h \in \widetilde{H}$ into $\boldsymbol{h}^\star = \boldsymbol{h}_\circ^\star, (\boldsymbol{h}_{\pi'}^\star)_{\pi' \in \mathsf{match}(\pi)}, \boldsymbol{h}_\triangle^\star$ as follows:

    - Lets $\ell \in \mathbb{L}$ be the label corresponding to $h = (r, \ell)$.

- For every $\pi' \in \mathsf{match}(\pi)$, samples random strings

$$\boldsymbol{r}^\star_{\pi'} := \left(r^\star_{\pi',\frac{1}{\alpha},L}, \quad r^\star_{\pi',\frac{1}{\beta},L}, \quad r^\star_{\pi',\frac{1}{\gamma},R}, \quad r^\star_{\pi',\frac{1}{\delta},R} \right) \quad ,$$

and uses them to construct the corresponding handles all relative to the label $\ell_{\mathcal{B}}$

$$\boldsymbol{h}^\star_{\pi'} := \left(h^\star_{\pi',\frac{1}{\alpha},L}, \quad h^\star_{\pi',\frac{1}{\beta},L}, \quad h^\star_{\pi',\frac{1}{\gamma},R}, \quad h^\star_{\pi',\frac{1}{\delta},R} \right) \quad .$$

- Constructs similarly the handles

$$\boldsymbol{h}^\star_\circ := \left(h^\star_{\circ,\alpha,L}, \quad h^\star_{\circ',\beta,R}, \quad h^\star_{\circ,\gamma,L}, \quad h^\star_{\circ,\delta,R} \right) \quad ,$$

- If $V(\{\ell\}) = \mathtt{true}$, constructs similarly the handles

$$\boldsymbol{h}^\star_\triangle = \left(h^\star_{\triangle,\eta,L}, \quad h^\star_{\triangle,\frac{1}{\eta}}, \quad h^\star_{\triangle,\varphi,R}, \quad h^\star_{\triangle,\frac{1}{\varphi}} \right) \quad .$$

2. Stores the mapping between each original handle h and corresponding generated handles $\boldsymbol{h}^\star$.
3. For each $h_i = (r_i, \ell_i) \in \widetilde{H}$, the simulator samples a random $\xi_{i,L} \leftarrow \mathbb{F}$, encodes $(\xi_{i,L}, \ell_i)$, and obtains a corresponding handle $h_{i,L}$.
4. Starts emulating $\mathcal{A}(\widetilde{Z}, \widetilde{H}^\star)$, where $\widetilde{H}^\star = \{\boldsymbol{h}^\star_1, \ldots, \boldsymbol{h}^\star_m\}$ are the handles constructed above.
5. Whenever $\mathcal{A}$ makes a zero-test query $(p^\star, \boldsymbol{h}^\star)$ meant for $\mathcal{B}^2$, the simulator $\mathcal{S}$:

- Writes $p^\star$ as a formal polynomial:

$$p^\star(\boldsymbol{h}^\star) = g^\star(\boldsymbol{h}^\star) + f^\star(\boldsymbol{h}^\star) \quad ,$$

where $g^\star(\boldsymbol{h}^\star)$ gathers the following expressions (the valid part of the polynomial)

$$\begin{aligned} g^\star(\boldsymbol{h}^\star) :=& \sigma + \sum_k \rho^L_k h^\star_{k,\triangle,\eta,L} h^\star_{k,\triangle,\frac{1}{\eta}} + \rho^R_k h^\star_{k,\triangle,\varphi,R} h^\star_{k,\triangle,\frac{1}{\varphi}} + \\ & \sum_{i \le j} \rho^{L,L}_{i,j} h^\star_{i,\circ,\alpha,L} h^\star_{j,\pi_i,\frac{1}{\alpha},L} + \rho^{L,R}_{i,j} h^\star_{i,\circ,\gamma,L} h^\star_{j,\pi_i,\frac{1}{\gamma},R} + \rho^{R,L}_{i,j} h^\star_{i,\circ,\beta,R} h^\star_{j,\pi_i,\frac{1}{\beta},L} + \rho^{R,R}_{i,j} h^\star_{i,\circ,\delta,R} h^\star_{j,\pi_i,\frac{1}{\delta},R} \quad , \end{aligned}$$

where $\pi_i = \Pi(\{\ell_i\})$.

- If $f^\star \not\equiv 0$, returns $\mathtt{false}$.
- In case $f^\star \equiv 0$, defines a new polynomial in $\boldsymbol{h}$:

$$\begin{aligned} g(\boldsymbol{h}) :=& \sigma + \sum_k \rho^L_k h_{k,L} + \rho^R_k \cdot (h_k - h_{k,L}) + \\ & \sum_{i \le j} \rho^{L,L}_{i,j} h_{i,L} h_{j,L} + \rho^{L,R}_{i,j} h_{i,L} \cdot (h_j - h_{j,L}) + \rho^{R,L}_{i,j} \cdot (h_i - h_{i,L}) \cdot h_{j,L} + \rho^{R,R}_{i,j} \cdot (h_i - h_{i,L}) \cdot (h_j - h_{j,L}) \quad . \end{aligned}$$

Submits the zero-test query $(g, \boldsymbol{h})$ to $\mathcal{M}^2$, and returns the answer.

We now prove that the statistical distance between

$$Z, \mathcal{S}^{\mathcal{M}^2,\mathcal{A}^{(\cdot)}}(\widetilde{Z},\widetilde{H}) \quad \text{and} \quad Z, \mathcal{A}^{\mathcal{B}^2}(\widetilde{Z},\widetilde{H}^\star) \ .$$

is at most $\mathrm{poly}(\lambda)/|\mathbb{F}| = \mathrm{negl}(\lambda)$.

We first note that the handles $\widetilde{H}^\star$ simulated by $\mathcal{S}$ are distributed identically to those given to $\mathcal{A}$ in the real experiment. Thus, it suffices to show that for every fixed zero test $p^\star(\boldsymbol{h}^\star) = g^\star(\boldsymbol{h}^\star) + f^\star(\boldsymbol{h}^\star)$, the answers simulated by $\mathcal{S}$ are statistically close to how real answers would be generated. For the rest of the argument, fix the underlying encoded pairs $(\xi_1,\ell_1),\dots,(\xi_m,\ell_m)$, and fix any polynomial $p^\star$ in $\boldsymbol{h}_1^\star,\dots,\boldsymbol{h}_m^\star$.

Case 1: $f^\star \equiv 0$**.** We consider first the case that $f^\star \equiv 0$, namely $p^\star \equiv g^\star$. We note that every monomial in $g^\star$ necessarily corresponds to "a valid product". Concretely, monomials $h^\star_{i,\circ,\cdot} h^\star_{j,\pi_i,\cdot}$ correspod to labels ℓ_i,ℓ_j with projections π_i,π_j such that $V(\{\ell_i,\ell_j\}) = V_\Pi(\pi_i,\pi_j) = \mathtt{true}$. Monomials $h^\star_{k,\triangle,\cdot} h^\star_{k,\triangle,\cdot}$ correspond to a label ℓ_k such that $V(\{\ell_k\}) = \mathtt{true}$. In this case, in the real experiment the oracle returns $\mathtt{true}$ if and only if

$$\begin{aligned}
0 = g^\star(\boldsymbol{\xi}) = \\
&\sigma + \sum_k \rho_k^L \xi_{k,L} + \rho_k^R \cdot \xi_{k,R} + \\
&\sum_{i\le j} \rho_{i,j}^{L,L}\xi_{i,L}\xi_{j,L} + \rho_{i,j}^{L,R}\xi_{i,L}\cdot\xi_{j,R} + \rho_{i,j}^{R,L}\cdot\xi_{i,R}\cdot\xi_{j,L} + \rho_{i,j}^{R,R}\cdot\xi_{i,R}\cdot\xi_{j,R} = \\
&\sigma + \sum_k \rho_k^L \xi_{k,L} + \rho_k^R \cdot (\xi_k - \xi_{k,L}) + \\
&\sum_{i\le j} \rho_{i,j}^{L,L}\xi_{i,L}\xi_{j,L} + \rho_{i,j}^{L,R}\xi_{i,L}\cdot(\xi_j - \xi_{j,L}) + \rho_{i,j}^{R,L}\cdot(\xi_i - \xi_{i,L})\cdot\xi_{j,L} + \rho_{i,j}^{R,R}\cdot(\xi_i - \xi_{i,L})\cdot(\xi_j - \xi_{j,L}) \ ,
\end{aligned}$$

where for each ξ_i, the field elements $\xi_{i,L},\xi_{i,R}$ are its random secret shares.

Indeed, the output of $g^\star(\boldsymbol{\xi})$ in the real experiment is distributed identically to that of $g(\boldsymbol{\xi})$ in the ideal (simulated) experiment, where g is the polynomial derived by the simulator form $g^\star$.

Case 2: $f^\star \not\equiv 0$**.** In this case, the ideal experiment always returns `false` . We will show that except with probability $\mathrm{poly}(\lambda)/|\mathbb{F}|$ over the choice of secret encoding key K and randomness used for secret sharing, the real experiment also returns `false` . That is, mapping the handles $\boldsymbol{h}^\star$ to the corresponding field elements $\boldsymbol{\xi}^\star$, it is the cae that $p^\star(\boldsymbol{\xi}^\star) \neq 0$.

To see this, consider the following sets of formal variables:

$$\Lambda := \left\{\alpha_\pi, \frac{1}{\alpha_\pi}, \beta_\pi, \frac{1}{\beta_\pi}, \gamma_\pi, \frac{1}{\gamma_\pi}, \delta_\pi, \frac{1}{\delta_\pi}\right\}_{\pi\in\Gamma} \cup \left\{\eta_\ell, \frac{1}{\eta_\ell}, \varphi_\ell, \frac{1}{\varphi_\ell}\right\}_{\ell\in\mathbb{L}}, \qquad \Psi := \{\xi_{i,L},\xi_{i,R}\}_{i\in[m]} \ .$$

Then mapping $\boldsymbol{h}^\star$ to the corresponding variables in Λ and Ψ, we can write $p^\star(\boldsymbol{\xi}^\star), f^\star(\boldsymbol{\xi}^\star), g^\star(\boldsymbol{\xi}^\star)$ as polynomials $P^\star(\Lambda,\Psi), F^\star(\Lambda,\Psi), G^\star(\Lambda,\Psi)$ in the variables of Λ and Ψ. To prove that $p(\boldsymbol{\xi}^\star) \neq 0$ with overwhelming probability, we will prove that $P^\star(\Lambda,\Psi) = F^\star(\Lambda,\Psi) + G^\star(\Lambda,\Psi) \neq 0$, with overwhelming probability. For this, we stratify:

$$F^\star(\Lambda,\Psi) = \sum_{\substack{x\in\Lambda \\ y\in\Lambda\cup\{1\}\setminus\{\frac{1}{x}\}}} Q_{x,y}(\Psi)\cdot x\cdot y \ ,$$

where every $Q_{x,y}(\Psi)$ is a polynomial of degree at most 2.

Since $f^\star \not\equiv 0$, there is some $Q_{x,y}$ that is not identically zero. We will first show that except with probability $2/|\mathbb{F}|$, over the choice of Ψ, all non-trivial polynomials $Q_{x,y}$ do not vanish. Then, fixing any Ψ

such that these do not vanish, we show that $P(\Lambda, \Psi)$ does not vanish except with probability $\mathrm{poly}(\lambda)/|\mathbb{F}|$ over the choice of Λ.

Indeed, fix any $Q_{x,y}(\Psi) \not\equiv 0$. If $Q_{x,y}(\Psi) \equiv c$, for some constant $c \in \mathbb{F} \setminus \{0\}$ which can occur for instance when $x, y \in \left\{\frac{1}{\eta_\ell}, \frac{1}{\varphi_\ell}\right\}$), then we are done. Thus, from hereon, we assume the existence of non-trivial monomials in Ψ.

Examining the monomials of $Q_{x,y}$, let us focus on the shares $\xi_{i,L}, \xi_{i,R}$ of ξ_i for some specific $i \in [m]$ such that $Q_{x,y}$ has non-trivial monomials in the variables $\xi_{i,L}, \xi_{i,R}$. We argue

Claim 4.2. *For any $w_L, w_R \in \Psi \cup \{1\}$, if the polynomial $Q_{x,y}$ includes both monomials $w_L \cdot \xi_{i,L}$ and $w_R \cdot \xi_{i,R}$, it must be that $w_L \neq w_R$.*

Proof. Throughout, we rely on the fact that all encodings are structured so that for any $k, j \in [m]$ the corresponding shares $\xi_{k,L}$ $\xi_{j,R}$ are never multiplied by the same scalar in Λ. In particular, we can assume from hereon that $x \neq y$, and that the monomials $w_L \cdot \xi_{i,L}$ and $w_R \cdot \xi_{i,R}$ originated from encodings of the form:

$$x \cdot \xi_{i,L}, \quad y \cdot \xi_{i,R}, \quad y \cdot w_L, \quad x \cdot w_R \ .$$

To rule out that $w_L = w_R$, we rule out the following possible cases:

- $w_L = w_R \in \Psi$. Assume w.l.o.g that $w_L = w_R = \xi_{k,L}$ for some k (the proof for $\xi_{k,R}$ is symmetric). In this case, the above yields an encoding of $x \cdot \xi_{k,R}$, which is a contradiction, since there already exists an encoding $x \cdot \xi_{i,L}$, and $\xi_{i,L}, \xi_{k,R}$ are never multiplied by the same scalar in Λ.
- $w_L = w_R = 1$. In this case, it must be that $\{x, y\} \subseteq \left\{\frac{1}{\eta_\ell}, \frac{1}{\varphi_\ell}\right\}_{\ell \in \mathbb{L}}$, since these are the only scalars encoded. This is again a contradiction since then there should not exist encodings of the form $x \cdot \xi_{i,L}, y \cdot \xi_{i,R}$.

□

Replacing all $\xi_{j,R}$ with $\xi_j - \xi_{j,L}$, and relying on the above Claim 4.2, we have that $Q_{x,y}(\Psi)$ is a non-zero polynomial in some monomial that includes some $\xi_{j,L}$. It follows, by Schwartz-Zippel (Fact 2.2, the simple version) that $Q_{x,y}(\Psi)$ vanishes with probability at most $2/|\mathbb{F}|$ over the choice of Ψ.

Next, fixing any such Ψ, and using the fact that $G^\star(\Lambda, \Psi)$ has no monomials xy such that $x \in \Lambda, y \in \Lambda \cup \{1\} \setminus \left\{\frac{1}{x}\right\}$, we now know that $P^\star(\Lambda, \Psi)$ is a non-zero polynomial in the variables Λ. By Schwartz-Zippel (Fact 2.2, the extended version), $P^\star$ vanishes with probability at most $\mathrm{poly}(\lambda)/|\mathbb{F}|$ over the choice of variables in Λ. Overall, $F^\star$ vanishes with probability at most $\mathrm{poly}(\lambda)/|\mathbb{F}|$.

This concludes the proof of the Lemma 4.3. □

4.3.2 From XIO with Oracle $\mathcal{M}^2$ to XIO with Oracle $\mathcal{B}^2$

We now deduce that any XIO scheme with explicit handles relative to $\mathcal{M}^2$ can be converted into a scheme relative to $\mathcal{B}^2$ (also with explicit handles).

Lemma 4.4. *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{(\cdot)}, \mathsf{xiO.Eval}^{(\cdot)})$ *be an XIO scheme, for a collection of circuit classes $\mathcal{C}$, defined relative to the (asymmetric) decomposable oracle $\mathcal{M}^2$, with explicit handles in $(\mathcal{X}, \mathcal{Y})$-product form, for some product collection $(\mathcal{X}, \mathcal{Y})$. Then* xiO *can be converted to a new scheme* $\mathsf{xiO}^\star$ *relative to the (symmetric) oracle $\mathcal{B}^2$, also with explicit handles in $(\mathcal{X}, \mathcal{Y})$-product form.*

Proof. We describe the new scheme.

The Obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}$**:** Given a circuit $C \in \boldsymbol{\mathcal{C}}$ with input size n, and security parameter 1^λ, and oracle access to $\mathcal{B}^2$, $\mathsf{xiO}^\star.\mathsf{Obf}^{\mathcal{B}^2}(C, 1^\lambda)$ does the following:

- **Emulate Obfuscation:**
  - Emulate $\mathsf{xiO}.\mathsf{Obf}^{\mathcal{M}^2}(C, 1^\lambda)$.
  - Throughout the emulation, store a list $L = \{(h, \xi)\}$ of encoded element-label pairs (ξ, ℓ) and corresponding handles $h = (r, \ell)$.
  - Obtain the obfuscation $\left(\widetilde{Z}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_X\right\}_{Y \in \mathcal{Y}_n}\right)$.
- **Recode Explicit Handles:**
  - Sample an encoding key K for $\mathcal{E}$. (If needed, K is sampled using lazy sampling as per remark 4.3.)
  - For each $X \in \mathcal{X}_n$:
    1. Retrieve $\widetilde{H}_X = (h_1, \dots, h_m)$ and the corresponding field elements and labels $(\xi_1, \ell_1), \dots, (\xi_m, \ell_m)$ from the stored list L.
    2. For each (ξ_i, ℓ_i), reencode $\mathcal{E}^{\mathcal{B}^2}((\xi_i, \ell_i), K)$ and obtain a handle $\boldsymbol{h}_i^\star$.
    3. Store $\widetilde{H}_X^\star = \{(h_i, \boldsymbol{h}_i^\star)\}_{i \in [m]}$.
  - For each $Y \in \mathcal{Y}_n$, symmetrically perform the above two steps with respect to $\widetilde{H}_Y$ (instead of $\widetilde{H}_X$) and store $\widetilde{H}_Y^\star$.
- **Output:**
  - $\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}_X^\star\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y^\star\right\}_{Y \in \mathcal{Y}_n}\right)$, where $\widetilde{C} := \left(\widetilde{Z}, \left\{\widetilde{H}_X\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y\right\}_{Y \in \mathcal{Y}_n}\right)$.

The Evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$**:** Given an obfuscation $\widetilde{C}^\star = \left(\widetilde{C}, \left\{\widetilde{H}_X^\star\right\}_{X \in \mathcal{X}_n}, \left\{\widetilde{H}_Y^\star\right\}_{Y \in \mathcal{Y}_n}\right)$, input $(x, y) \in \boldsymbol{X}_n \times \boldsymbol{Y}_n$, and oracle access to $\mathcal{M}^2$, the evaluator $\mathsf{xiO}^\star.\mathsf{Eval}^{\mathcal{B}^2}(\widetilde{C}^\star, (x, y))$ does the following:

- Emulate $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{M}^2}(\widetilde{C}, (x, y))$.
- Emulate any zero-test query $(p, h_1, \dots, h_m)$ it makes to $\mathcal{M}^2$ as follows:
  1. Let $(X, Y) \in \mathcal{X}_n \times \mathcal{Y}_n$ be the (unique) sets such that $(x, y) \in X \times Y$. Retrieve $\widetilde{H}_X, \widetilde{H}_Y$.
  2. Obtain the handles $\boldsymbol{h}_1^\star, \dots, \boldsymbol{h}_m^\star$ corresponding to $h_1, \dots, h_m$ from $\widetilde{H}_X^\star \cup \widetilde{H}_Y^\star$.
  3. Run the decoder $\mathcal{D}^{\mathcal{B}^2}(p, \boldsymbol{h}_1^\star, \dots, \boldsymbol{h}_m^\star)$.

Clearly the new scheme also has explicit handles in product form (with respect to the same product collection). The size of each set of handles now blows up by a factor of $\mathrm{poly}(\lambda)$, as $\mathcal{E}$ translates any handle h_i to a list of handles $\boldsymbol{h}_i^\star$ including $O(\mathsf{Arity}(V^2))$ handles, where the arity of the validity predicate V^2 of $\mathcal{M}^2$ is bounded by $\mathrm{poly}(\lambda)$.

The simulator guaranteed by Lemma 4.3 directly implies that if the original scheme satisfies indistinguishability so does the new scheme. We show that it also implies that the new scheme is as correct as the original one. Consider, the adversary $\mathcal{A}^{\mathcal{B}^2}(\widetilde{C}^\star)$ that simply samples a random input $u \leftarrow \{0,1\}^n$, runs the evaluation algorithm $\mathsf{xiO}^\star.\mathsf{Eval}^{\mathcal{B}^2}(\widetilde{C}^\star, u)$, obtains the result $v^\star$, and outputs $(u, v^\star)$. Observe that for this adverasry the simulator $\mathcal{S}^{\mathcal{M}^2,\mathcal{A}}(\widetilde{C})$, exactly emulates $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{M}^2}(\widetilde{C}, z)$, obtains the result v, and outputs (u, v). By the simulation guarantee (u, v) and $(u, v^\star)$ are identically distributed upto a small statistical difference, and thus the new scheme has the same (approximate) correctness as the original scheme up to a negligible difference. Looking closer, note that for the honestly evaluating adversary considered, the simulator in fact perfectly emulates evaluation without any statistcal error (corresponding to case 1 in the analysis).

□

4.4 Putting it All Together

We now conclude Theorem 4.1. Then we explain how to instantiate the symmetric bilinear oracle used by the XIO scheme produced by the transformation of Theorem 4.1 with actual bilinear pairing groups.

4.4.1 Concluding Theorem 4.1

We, in fact, prove a stronger statement than the one stated in the beginning of the section, in which *the resulting* $\mathsf{xiO.Obf}$ *scheme is also with explicit handles in product form.*

Theorem 4.1 (Restated)**.** *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{M}}, \mathsf{xiO.Eval}^{\mathcal{M}})$ *be an* $\mathsf{xiO.Obf}$ *scheme, relative to a degree-d decomposable ideal graded encoding oracle $\mathcal{M}$, for a collection of circuit classes $\mathcal{C}$ that is in $(\mathcal{X}, \mathcal{Y})$-product form, for some product collection $(\mathcal{X}, \mathcal{Y})$. Further assume that for some constant $\gamma < 1$,*

$$|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}} \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right)\right)^d + |\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}} \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right)\right)^d \leq 2^{\gamma n} \cdot \mathrm{poly}(|C|, \lambda) \ .$$

Then xiO *can be converted into an approximately-correct scheme* $\mathsf{xiO}^\star$ *relative to the symmetric bilinear oracle $\mathcal{B}^2$, with explicit handles in $(\mathcal{X}, \mathcal{Y})$-product form.*

Proof of Theorem 4.1. To obtain $\mathsf{xiO}^\star$, we apply to xiO Lemmas 4.1, 4.2, 4.4.

- Lemma 4.1 turns xiO into an approximately-correct XIO scheme xiO_1 with explicit handles, relative to the same degree-d decomposable oracle $\mathcal{M}^d$ that xiO uses.
- Lemma 4.2 turns xiO_1 into an approximately-correct XiO scheme xiO_2 with explicit handles, relative to an asymmetric bilinear oracle $\mathcal{M}^2$ that is also decomposable.
- Lemma 4.4 turns xiO_2 into an approximately-correct XiO scheme xiO_3 with explicit handles, relative to a symmetric bilinear oracle $\mathcal{B}^2$.

The final XIO scheme xiO_3 is exactly the new XIO scheme $\mathsf{xiO}^\star$. By composing the three lemmas, we have that $\mathsf{xiO}^\star$ is approximately correct and secure. The only thing left to argue that $\mathsf{xiO}^\star$ is also weakly succinct. Recall that any obfuscated circuit in $\mathsf{xiO}^\star$ has the form

$$\widetilde{C} = \left(\widetilde{Z}, \{\widetilde{H}_X\}, \{\widetilde{H}_Y\},\ \{\widetilde{H}^\star_X\}, \{\widetilde{H}^\star_Y\},\ \{\widetilde{H}^{\star\star}_X\}, \{\widetilde{H}^{\star\star}_Y\}\right)$$

where $\widetilde{Z}$ is an obufscated circuit of the original scheme xiO, $\widetilde{H}_X$ and $\widetilde{H}_Y$ are the sets of explicit handles of $\mathcal{M}^d$ added by Lemma 4.1, $\widetilde{H}^\star_X$ and $\widetilde{H}^\star_Y$ are the encodings of monomials of $\mathcal{M}^2$ added by Lemma 4.2, $\widetilde{H}^{\star\star}_X$

and $\widetilde{H}_Y^{\star\star}$ are the re-encodings of $\mathcal{B}^2$ added by Lemma 4.4. By the three lemmas and the fact that the original scheme xiO is α-compressing, for some $\alpha < 1$, and satisfies the bound required by Theorem 4.1, we have,

$$\begin{aligned}
|\widetilde{C}| &\leq |\widetilde{Z}| + O\left(\left|\left\{\widetilde{H}_X^{\star\star}\right\}, \left\{\widetilde{H}_Y^{\star\star}\right\}\right|\right) \\
&\leq 2^{\alpha n}\text{poly}(\lambda, |C|) \\
&\quad + \left(|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}} \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right)\right)^d + |\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}} \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right)\right)^d\right) \cdot \text{poly}(\lambda) \\
&\leq (2^{\alpha n} + 2^{\gamma n}) \cdot \text{poly}(\lambda, |C|) \\
&\leq 2^{\beta n} \cdot \text{poly}(\lambda, |C|) \ ,
\end{aligned}$$

for some $\beta = \max(\alpha, \gamma) < 1$. Thus, the new XIO scheme is weakly succinct as required. □

4.4.2 Instantiation in Bilinear Pairing Groups

Given the final XIO scheme xiO relative to the ideal symmetric bilinear oracle $\mathcal{B}^2$, we would like to instantiate the oracle with actual bilinear pairing groups (G, G_T, g, g_T, e) to obtain a scheme in the plain model. However, the ideal oracle $\mathcal{B}^2$ does not completely align with traditional instantiations of bilinear groups (based on elliptic curves), with two notable differences. First, $\mathcal{B}^2$ generates a new randomized encodings for every field element ξ, whereas in bilinear pairing groups all encodings of ξ have a unique form g^ξ. Second, $\mathcal{B}^2$ only supports zero testing but not homomorphic operations, whereas one can homomorphically add encodings in bilinear pairing groups G, G_T and homomorphically multiply encodings in G.

We next sketch how xiO relative to $\mathcal{B}^2$ can be transformed into a scheme xiO* relative to another ideal oracle $\mathcal{B}^{\star 2}$ that captures standard instantiations of bilinear groups, aligning with standard formulation of *the generic bilinear group model* [Sho97].

The Oracle $\mathcal{B}^{\star 2}$. In $\mathcal{B}^{\star 2}$ elements may be associated with one of two labels ℓ_B and ℓ_T representing the base and target groups, respectively. The oracle $\mathcal{B}^{\star 2}$ associates with any single field element ξ and handle $\ell \in \{\ell_B, \ell_T\}$ a unique handle. Given handles for field elements ξ, ξ' with the same label (ℓ_B or ℓ_T), the oracle $\mathcal{B}^{\star 2}$ allows obtaining a handle for their sum $\xi + \xi'$ under that label. Given handles for field elements ξ, ξ' both under label ℓ_B, $\mathcal{B}^{\star 2}$ allows obtaining a handle for their product $\xi \cdot \xi'$ under label ℓ_T. The oracle permits degree-two zero tests $p(\boldsymbol{\xi})$ if all elements ξ_i are encoded under ℓ_B. Otherwise (some element is encoded under the target group label ℓ_T), p is valid only if it is of degree one.

The New Scheme xiO*. Recall that, by Theorem 4.1, xiO has explicit handles. The new scheme xiO* obfuscates any circuit C as follows:

- Run the obfuscator using $\mathsf{xiO}^{(\cdot)}(C, 1^\lambda)$, emulating the oracle $\mathcal{B}^2$ internally. Store a list $L = \{(h, \xi)\}$ of encoded element-label pairs (ξ, ℓ) and corresponding handles $h = (r, \ell)$.
- Obtain an obfuscation $\widetilde{C} = (Z, \boldsymbol{h})$, where $\boldsymbol{h}$ is the list of explicit handles.
- For every explicit handle $h_i \in \boldsymbol{h}$, obtain the corresponding field element ξ_i from the list L, and encode it using the oracle $\mathcal{B}^{\star 2}$ under the (base-group) label ℓ_B, obtaining a (unique) handle $h_i^\star$.
- Output as the obfuscated circuit $\widetilde{C}^\star = (\widetilde{C}, \boldsymbol{h}^\star)$.

The obfuscated circuit $\widetilde{C}^\star$ is evaluated on any input x, by emulating an evaluation of $\widetilde{C}$, translating any zero-test query on $\boldsymbol{h}$ to a zero-test query on $\boldsymbol{h}^\star$, with respect to the same polynomial. The correctness and succinctness of the new scheme follow readily.

Security. To show that $\mathsf{xiO}^\star$ is as secure as xiO, we show that there exists a simualtor $\mathcal{S}$ that with access to $\mathcal{B}^2$ and after receiving an obfuscated circuit $\widetilde{C}$ under xiO with oracle $\mathcal{B}^2$, can emulate the view of any attacker $\mathcal{A}$ that has access to $\mathcal{B}^{\star 2}$ and receives an obfuscated circuit $\widetilde{C}^\star$ under $\mathsf{xiO}^\star$ with oracle $\mathcal{B}^{\star 2}$.

The simulator $\mathcal{S}^{\mathcal{B}^2}(\widetilde{C})$ proceeds as follows:

- Parse $\widetilde{C} = (Z, \boldsymbol{h})$, where $\boldsymbol{h} = h_1, \cdots h_m$ are the explicit handles relative to $\mathcal{B}^2$.
- To generate $\widetilde{C}^\star$ for $\mathcal{A}$, for every pair of handles h_i, h_j in $\boldsymbol{h}$, $\mathcal{S}$ tests whether they encode the same value making a zero test to its oracle $\mathcal{B}^2$ for the degree-one polynomial $p(h_i, h_j) = h_i - h_j$. $\mathcal{S}$ records all "equality relations" $h_i \sim h_j$, and generates new handles $h_1^\star, \cdots, h_m^\star$ subject to the equality relation, that is, $h_i^\star = h_j^\star$ if $h_i \sim h_j$. $\mathcal{S}$ gives $\mathcal{A}$ the obfuscated circuit $\widetilde{C}^\star = (\widetilde{C}, \boldsymbol{h}^\star)$, and starts emulating it.
- Throughout, $\mathcal{S}$ keeps a list L, consisting of every handle $h^\star$ it generates and a corresponding formal polynomial p over variables $\boldsymbol{h}$, such that, the value encoded in $h^\star$ equals to p evaluated on the values encoded in $\boldsymbol{h}$. L is initialized with the set of handles generated above $(h_i^\star, p_i(\boldsymbol{h}) = h_i)$.
- Whenever $\mathcal{A}$ asks $\mathcal{B}^{\star 2}$ to encode an element ξ under label $\ell \in \{\ell_B, \ell_T\}$, S tests whether v equals to the value encoded in any handle $(h^\star, p) \in L$ with the same label ℓ. To do so, it zero tests whether $p(\boldsymbol{h}) - v$ is zero using its own oracle $\mathcal{B}^2$. If such a tuple exists, $\mathcal{S}$ simply returns $h^\star$; otherwise, it returns a new fresh handle $h^\star$ and adds to L the tuple $(h^\star, p(\boldsymbol{h}) \equiv v)$.
- Whenever $\mathcal{A}$ asks $\mathcal{B}^{\star 2}$ to homomorphically add/multiply two handles $h_i^\star, h_j^\star$, $\mathcal{S}$ finds the corresponding tuples $(h_i^\star, p_i)$ and $(h_j^\star, p_j)$ in L, and verifies that the homomorphic operation is allowed (both labels corresponding to the two are equal $\ell_i = \ell_j$, and if its multiplication it further holds that $\ell_i = \ell_j = \ell_B$). Let ℓ be the output label ($\ell = \ell_i = \ell_j$ for addition, and $\ell = \ell_T$ for multiplocation). $\mathcal{S}$ tests whether the result of the opertation equals to the value encoded in any handle $(h^\star, p) \in L$ with label ℓ. This is done by zero testing whether $p(\boldsymbol{h}) - (p_i(\boldsymbol{h}) \ + / \times \ p_j(\boldsymbol{h}))$ is zero using $\mathcal{B}^2$. If such a tuple exists, simply return $h^\star$; otherwise, return a fresh handle $h^\star$ and add to L the tuple $(h^\star, p(\boldsymbol{h}) = p_i(\boldsymbol{h}) \ + / \times \ p_j(\boldsymbol{h}))$.
- Whenever $\mathcal{A}$ asks $\mathcal{B}^{\star 2}$ to zero test $(p, h_{i_1}^\star, \cdots, h_{i_k}^\star)$, $\mathcal{S}$ finds all corresponding tuples $(h_{i_j}^\star, p_{i_j})$ in L, and verifies that p is valid. Then, it answers the query by zero-testing whether $p(p_{i_1}(\boldsymbol{h}), \cdots, p_{i_k}(\boldsymbol{h}))$ is zero using $\mathcal{B}^2$.

It is not hard to check that $\sim$ defined as above perfectly emulates the view of $\mathcal{A}$. Thus, if xiO, relative to $\mathcal{B}^2$, is secure, $\mathsf{xiO}^\star$, relative to $\widetilde{\mathcal{B}}^2$, is secure.

Assuming Über Security. The oracle $\mathcal{B}^{\star 2}$ captures traditional bilinear pairing groups, and can accordingly be instantiated in such groups. Correctness and succinctness follows immediately. For security to hold, we assume a strong *über* assumption [BBG05, Boy08] on symmetric bilinear pairing groups.

Roughly speaking, the über assumption states that encodings of two sets of elements under the same set of labels are indistinguishable if no efficient attacker can tell apart ideal encodings of these two sets of values under the same labels. In essence, under the über assumption, one can translate security in the ideal model to security in the plain model. By making the über assumption on symmetric bilinear pairing groups, the approximate XIO scheme $\mathsf{xiO}^\star$ above is secure when instantiated in such groups.

5 Removing Constant-Degree Oracles from Sufficiently-Compressing XIO

We show that any XIO scheme relative to a degree-d ideal oracle $\mathcal{M}^d$, with obfuscation-evaluation query complexity (q_o, q_e) such that $q_o^d \cdot q_e$ is polynomially smaller than the input space, can be transformed into an approximately-correct XIO scheme in the plain model (i.e., without any oracles). For $d = 1$, which is analogous to the generic-group model [Sho97] (for the multiplicative group of a field), we show in Section 7.2 that any unbounded-key functional encryption scheme implies an XIO scheme as above. For $d \geq 2$, we do not know how to obtain such XIO schemes from functional encryption. (Interestingly, a non-black-box constructions of XIO with arbitrary constant compression is known from unbounded-key functional encryption [BNPW16].)

Theorem 5.1. *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{M}}, \mathsf{xiO.Eval}^{\mathcal{M}})$ *be an* $\mathsf{xiO.Obf}$ *scheme, relative to a degree-d ideal graded encoding oracle* $\mathcal{M}^d$*, for a collection of circuit classes* $\boldsymbol{\mathcal{C}}$ *such that for some constant* $\gamma < 1$*,* $q_o^d \cdot q_e \leq 2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|)$*. Then* xiO *can be converted into a new approximately-correct scheme* $\mathsf{xiO}^\star$ *in the plain model.*

Putting aside the efficiency requirement of XIO, the transformation is similar to that of Pass and shelat [PS16]. We choose our parameters carefully to obtain the desired XIO compression. We also give a different, relatively simple, correctness proof.

Proof of Theorem 5.1. In what follows, let $\mathsf{xiO} = (\mathsf{xiO.Obf}^{(\cdot)}, \mathsf{xiO.Eval}^{(\cdot)})$ be an XIO scheme relative to a degree-d oracle $\mathcal{M}^d$ for a collection of circuit classes $\boldsymbol{\mathcal{C}} = \{\mathcal{C} = \{\mathcal{C}_\lambda\}\}$ with query complexity (q_o, q_e). We construct a new obfuscator $\mathsf{xiO}^\star = (\mathsf{xiO}^\star.\mathsf{Obf}, \mathsf{xiO}^\star.\mathsf{Eval})$ for $\mathcal{C}$ in the plain model. As in Section 4, given an oracle $\mathcal{M}^d$, and a zero-test $(p, \boldsymbol{h})$ made to the oracle, we will say that it is a *true zero-test* if $\mathcal{M}^d(p, \boldsymbol{h}) = \mathtt{true}$; namely, it is valid and evaluates to zero on the corresponding field elements.

Without Loss of Generality. We make the same w.l.o.g assumptions as in Section 4; namely, that the obfuscator only encodes and does not zero-test, and that the evaluator and the adversary only zero-test and do not encode.

The Obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}$. Given a circuit $C \in \boldsymbol{\mathcal{C}}$ with input size n, and security parameter 1^λ, $\mathsf{xiO}^\star.\mathsf{Obf}$ does the following:

- **Emulate Obfuscation:**
  - Emulate $\mathsf{xiO.Obf}^{\mathcal{M}^d}(C, 1^\lambda)$.
  - Throughout the emulation, emulate the oracle $\mathcal{M}^d$, storing a list $L = \{(h, \xi)\}$ of encoded element-label pairs (ξ, ℓ) and corresponding handles $h = (r, \ell)$.
  - Obtain the obfuscation $\widetilde{C}$.
- **Learn Heavy Subspace:** Let $\mathcal{P} = \emptyset$. Repeat the following for $100 \cdot (q_o + 1)^d$ iterations:
  - Sample a random input $x_i \leftarrow \{0,1\}^n$.
  - Emulate $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x_i)$. To answer oracle calls, emulate $\mathcal{M}^d$ using the stored list L.
  - For any true zero-test $(p, \boldsymbol{h})$ — that is, a zero-test query that evaluates to $\mathtt{true}$ — if

$$p(\boldsymbol{h}) \notin \mathrm{span}(\mathcal{P}) := \left\{ \sum_i \rho_i p_i(\boldsymbol{h}_i) \;\middle|\; (p_i, \boldsymbol{h}_i) \in \mathcal{P}, \rho_i \in \mathbb{F} \right\} ,$$

    add $(p, \boldsymbol{h})$ to $\mathcal{P}$.
- **Output:** $\widetilde{C}^\star = (\widetilde{C}, \mathcal{P})$.

The Evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$**.** Given $(\widetilde{C}, \mathcal{P})$ and $x \in \{0,1\}^n$:

- Emulate the oracle-aided $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$. For any zero-test $(p, \boldsymbol{h})$, answer `true` if $p(\boldsymbol{h}) \in \mathrm{span}(\mathcal{P})$, and `false` otherwise.

Proposition 5.1. $\mathsf{xiO}^\star$ *has non-trivial efficiency, if* $q_o^d \cdot q_e \leq 2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|)$*, for some constant* $\gamma < 1$.

Proof. The size of the obfuscation $\widetilde{C}^\star$ is exactly the size of the oracle obfuscation $|\widetilde{C}|$ plus the size of the set of polynomials $\mathcal{P}$ recording during the learning of a heavy subspace. $\widetilde{C} = 2^{(1-\Omega(1))n} \cdot \mathrm{poly}(\lambda, |C|)$ by the non-trivial efficiency of the original scheme xiO. The size of $\mathcal{P}$ is bounded by $K \cdot q_e$, where K is the number of iterations the the learning step is performed. The number of iterations K is $100(q_o+1)^d$. It follows that

$$|\mathcal{P}| \leq O(K \cdot q_e) \leq q_o^d \cdot q_e \leq 2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|) \enspace ,$$

as required. □

Proposition 5.2. $\mathsf{xiO}^\star$ *is approximately-correct.*

Proof. We first note that $\mathrm{span}(\mathcal{P})$ is a subspace of the linear space $\mathcal{T}$ of all true zero tests over the set of all handles H created during obfuscation

$$\mathcal{T} := \left\{ (p, \boldsymbol{h}) \;\middle|\; \mathcal{M}^d(p, \boldsymbol{h}) = \mathtt{true} \right\} \enspace ,$$

where we naturally define linear operations for any $\boldsymbol{h}_0, \boldsymbol{h}_1 \subseteq H, p_0, p_1 \in \mathbb{F}[\boldsymbol{h}_0 \cup \boldsymbol{h}_1]$:

$$\rho_0 \cdot (p_0, \boldsymbol{h}_0) + \rho_1 \cdot (p_1, \boldsymbol{h}_1) = (\rho_0 \cdot p_0 + \rho_1 \cdot p_1, \boldsymbol{h}_0 \cup \boldsymbol{h}_1) \enspace .$$ [6]

The number of monomials in H is bounded by $(|H|+1)^d \leq (q_o+1)^d$, which gives a bound on the dimension of $\mathrm{span}(\mathcal{P})$.

Fix any circuit $C \in \mathcal{C}$ and security parameter $\lambda \in \mathbb{N}$. Recall that we would like to show that when sampling an obfuscation $(\widetilde{C}, \mathcal{P}) \leftarrow \mathsf{xiO}^\star.\mathsf{Obf}(C, 1^\lambda)$, and a random input $x \leftarrow \{0,1\}^n$, it is the case that $\mathsf{xiO}^\star.\mathsf{Eval}((\widetilde{C}, \mathcal{P}), x) = C(x)$, except with probability $1/100$. We shall denote this experiment by Exp.

To this end, we define an event Bad that captures when the result of evaluation is incorrect. The event Bad occurs in the experiment Exp if when the evaluator $\mathsf{xiO}^\star.\mathsf{Eval}((\widetilde{C}, \mathcal{P}), x)$ emulates the oracle-aided evaluator $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$, it is required to answer a zero test $(p, \boldsymbol{h}) \in \mathcal{T} \setminus \mathrm{span}(\mathcal{P})$. That is, a true zero test (relative to the oracle $\mathcal{M}^d$ in the emulated obfuscation) that increases the dimension of the set $\mathcal{P}$ generated during the learning step.

Indeed, as long as Bad does not occur the emulated evaluation $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$ is distributed exactly as it would had been if $\widetilde{C}$ were generated by $\mathsf{xiO.Obf}^{\mathcal{M}^d}(C, 1^\lambda)$ with the actual oracle $\mathcal{M}^d$ and $\mathsf{xiO.Eval}^{\mathcal{M}^d}(\widetilde{C}, x)$ would be performed relative to $\mathcal{M}^d$. Thus, it is sufficient to show that

$$\Pr_{\mathsf{Exp}}[\mathsf{Bad}] \leq 1/100 \enspace .$$

We claim that

$$\Pr[\mathsf{Bad}] \leq \frac{\mathbb{E}|\mathcal{P}|}{K} \leq \frac{(q_o+1)^d}{100(q_o+1)^d} \leq 1/100 \enspace .$$

[6]Note that these linear operations respect validity for any well-formed validity predicate.

The second inequality follows from the bound $(q_o + 1)^d$ on the dimension of $\mathcal{P}$, which implies that it grows at most $(q_o + 1)^d$ times. To see the first inequality, recall that Bad occurs when the evaluator makes a query $(p, \boldsymbol{h}) \in \mathcal{T} \setminus \mathrm{span}(\mathcal{P})$ that wasn't covered. In particular, for any step $i \in [K]$, the probability of adding a polynomial to $\mathcal{P}$ is at least $\Pr[\mathsf{Bad}]$, and the expected number of added polynomials is at least $K \cdot \Pr[\mathsf{Bad}]$. □

Proposition 5.3. $\mathsf{xiO}^\star$ *preserves the security security guarantee of* xiO.

Proof sketch. The new obfuscation $\widetilde{C}^\star$ consists exactly of an obfuscation $\widetilde{C}$ under the original scheme, plus the set $\mathcal{P}$, which was generated in the learning phase. However, the learning phase, simply consists of running the evaluation algorithm of the original scheme repeatedly for a polynomial number of times. Thus, any attacker against the new scheme can be perfectly simulated by an attacker for the original scheme that simulates the above handles by performing the learning process by itself. □

This completes the proof of Theorem 5.1.

□

6 Removing Random Oracles

We show that any XIO scheme in the random-oracle model, with obfuscation-evaluation query complexity (q_o, q_e) such that $q_o \cdot q_e$ is polynomially smaller than the input space, can be transformed into an approximately-correct XIO scheme in the plain model. In Section 7.2, we show that any unbounded-key functional encryption scheme in the random-oracle model implies such an XIO scheme.

Theorem 6.1. *Let* $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{R}}, \mathsf{xiO.Eval}^{\mathcal{R}})$ *be an* $\mathsf{xiO.Obf}$ *scheme, relative to a random oracle* $\mathcal{R}$*, for a collection of circuit classes* $\boldsymbol{\mathcal{C}}$ *such that for some constant* $\gamma < 1$*,* $q_o \cdot q_e \leq 2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|)$*. Then* xiO *can be converted into a new approximately-correct scheme* $\mathsf{xiO}^\star$ *in the plain model.*

Proof of Theorem 6.1. Putting aside the efficiency requirement of XIO, the transformation is similar to that of Canetti, Kalai, and Paneth [CKP15]. For the sake of completeness, and to address the XIO efficiency features, we describe the transformation in full. Our transformation also has slightly better parameters and a different correctness proof, simplifying the proof in [CKP15].

In what follows, let $\mathsf{xiO} = (\mathsf{xiO.Obf}^{(\cdot)}, \mathsf{xiO.Eval}^{(\cdot)})$ be an XIO scheme in the random-oracle model for a collection of circuit classes $\boldsymbol{\mathcal{C}} = \{\mathcal{C} = \{\mathcal{C}_\lambda\}\}$ with query complexity (q_o, q_e). We construct a new obfuscator $\mathsf{xiO}^\star = (\mathsf{xiO}^\star.\mathsf{Obf}, \mathsf{xiO}^\star.\mathsf{Eval})$ for $\mathcal{C}$ in the plain model.

The Obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}$**.** Given a circuit $C \in \boldsymbol{\mathcal{C}}$ with input size n, and security parameter 1^λ, $\mathsf{xiO}^\star.\mathsf{Obf}$ does the following:

- **Emulate Obfuscation:** Emulate the oracle-aided $\mathsf{xiO.Obf}^{(\cdot)}(C, 1^\lambda)$. To answer its oracle calls, emulate the random oracle $\mathcal{R}$ (using standard lazy sampling). Store the resulting obfuscation $\widetilde{C}$, and the partial random oracle $\mathcal{R}_{\mathbf{Q}}$ consisting the set $\mathbf{Q}$ of at most $q_o = q_o(C, \lambda)$ queries made during the execution along with their emulated answers.
- **Learn Heavy Queries:** Let $\mathcal{R}_0 = \emptyset$. For $i \in \{1, \ldots, K := 100q_o\}$ do the following,
  - Sample a random input $x_i \leftarrow \{0, 1\}^n$.

– Emulate $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x_i)$. To answer oracle calls, emulate the random oracle $\mathcal{R}$ (using standard lazy sampling) consistently with $\mathcal{R}_{\mathbf{Q}}$ and $\mathcal{R}_{i-1}$.
– Let $\mathcal{R}_i = \mathcal{R}_{i-1} \cup \mathcal{R}_{\mathbf{Q}_i}$ be the partial random oracle consistent with $\mathcal{R}_{i-1}$ and extended to include the set $\mathbf{Q}_i$ of q_e queries made in the current execution along with their emulated answers.

- **Output:** $\widetilde{C}^\star = (\widetilde{C}, \mathcal{R}_K)$.

The Evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$**.** Given $(\widetilde{C}, \mathcal{R}_K)$ and $x \in \{0,1\}^n$:

- Emulate the oracle-aided $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$. To answer oracle calls, emulate the random oracle $\mathcal{R}$ (using standard lazy sampling) consistently with $\mathcal{R}_K$.

Proposition 6.1. $\mathsf{xiO}^\star$ *has non-trivial efficiency, if* $q_o \cdot q_e \leq 2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|)$*, for some constant* $\gamma < 1$*.*

Proof. The size of the obfuscation $\widetilde{C}^\star$ is exactly the size of the random-oracle obfuscation $|\widetilde{C}|$ plus the size of queries and answers made by the evaluation algorithm during the learning of heavy queries $|\mathcal{R}_K|$. $\widetilde{C} = 2^{(1-\Omega(1))n} \cdot \mathrm{poly}(\lambda, |C|)$ by the non-trivial efficiency of the original scheme xiO, and the size of $\mathcal{R}_K$ is bounded by $K \cdot q_e = 100 q_o \cdot q_e$, corresponding to K executions of $\mathsf{xiO.Eval}$. □

Proposition 6.2. $\mathsf{xiO}^\star$ *is approximately-correct.*

The argument follows similar rationale as the argument behind Proposition 6.2 in Section 5, but is slightly more delicate. There the state of the oracle $\mathcal{M}^d$ can be fixed after the emulated obfuscation (as evaluation only zero-tests), whereas here the evaluator can query the random oracle on points that were previously not queried. We proceed to the formal argument.

Proof. Fix any circuit $C \in \mathcal{C}$ and security parameter $\lambda \in \mathbb{N}$. Recall that we would like to show that when sampling an obfuscation $(\widetilde{C}, \mathcal{R}_K) \leftarrow \mathsf{xiO}^\star.\mathsf{Obf}(C, 1^\lambda)$, and a random input $x \leftarrow \{0,1\}^n$, it is the case that $\mathsf{xiO}^\star.\mathsf{Eval}((\widetilde{C}, \mathcal{R}_K), x) = C(x)$, except with probability $1/100$. We shall denote this experiment by Exp.

To this end, we define an event Bad that captures when the result of evaluation is incorrect. The event Bad may occur in the experiment Exp only if when the evaluator $\mathsf{xiO}^\star.\mathsf{Eval}((\widetilde{C}, \mathcal{R}_K), x)$ emulates the oracle-aided evaluator $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$, it is required to answer an oracle query $Q \in \mathbf{Q} \setminus \cup_{i=1}^K \mathbf{Q}_i$. That is, a query that was performed during the obfuscation when emulating $\mathsf{xiO.Obf}^{(\cdot)}(C, 1^\lambda)$, but was not performed when learning heavy queries. We argue that, unless Bad occurs, the evaluation in Exp would be correct. Indeed, as long as Bad does not occur, the emulated evaluation $\mathsf{xiO.Eval}^{(\cdot)}(\widetilde{C}, x)$ is distributed exactly as it would have been if $\widetilde{C}$ were generated by $\mathsf{xiO.Obf}^{\mathcal{R}}(C, 1^\lambda)$ with a true random oracle $\mathcal{R}$ and $\mathsf{xiO.Eval}^{\mathcal{R}}(\widetilde{C}, x)$ would be performed relative to $\mathcal{R}$. Thus, it is sufficient to show that

$$\Pr_{\mathsf{Exp}}[\mathsf{Bad}] \leq 1/100 \ .$$

To bound the probability that Bad occurs, we consider an alternative (mental) experiment Exp' where obfuscation and a single evaluation are performed as follows:

- A full (rather than partial) random oracle $\mathcal{R}$ is sampled.

- The obfuscation $\mathsf{xiO}^\star.\mathsf{Obf}(C, 1^\lambda)$ is performed, only that instead of lazy sampling the partial oracles $(\mathcal{R}_{\mathbf{Q}}, \mathcal{R}_1, \ldots, \mathcal{R}_K)$, all oracle queries are answered according to $\mathcal{R}$. (We will still address the sets of queries $\mathbf{Q}, \mathbf{Q}_1, \ldots, \mathbf{Q}_K$, which are defined in the same way, but answered according to $\mathcal{R}$.)

- Then a single evaluation query is performed for a random input $x \leftarrow \{0,1\}^n$, again using $\mathcal{R}$ to answer all oracle queries.

We can accordingly define an event Bad in Exp' analogous to Bad in Exp. Namely, when the evaluator $\mathsf{xiO}^{\star}.\mathsf{Eval}((\widetilde{C}, \mathcal{R}_K), x)$ emulates the oracle-aided evaluator $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{R}}(\widetilde{C}, x)$, it makes an oracle query $Q \in \mathbf{Q} \setminus \cup_{i=1}^{K} \mathbf{Q}_i$. We next observe that until the point that Bad occurs in Exp or in Exp', respectively, the two experiments are identically distributed. Indeed, the only difference is that in Exp the random oracle is lazy sampled as the experiment progresses, rather than being sampled ahead of time as in Exp'. In particular, it holds that

$$\Pr_{\mathsf{Exp}}[\mathsf{Bad}] = \Pr_{\mathsf{Exp}'}[\mathsf{Bad}] \quad .$$

We now bound the probability that Bad occurs in Exp'. For this, fix the random oracle $\mathcal{R}$, fix the coins of the emulated obfuscation $\mathsf{xiO}.\mathsf{Obf}^{\mathcal{R}}(C, 1^{\lambda})$, and let $\mathbf{Q}$ be the corresponding set of queries made during the obfuscation. For any $Q \in \mathbf{Q}$, we let Bad_Q be the event that $Q \notin \cup_{i=1}^{K} \mathbf{Q}_i$, but $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{R}}(\widetilde{C}, x)$ makes the oracle Q. Then, by a union bound

$$\Pr_{\mathsf{Exp}'}[\mathsf{Bad}] \le \sum_{Q \in \mathbf{Q}} \Pr_{\mathsf{Exp}'}[\mathsf{Bad}_Q] \le q_o \cdot \max_{Q \in \mathbf{Q}} \Pr_{\mathsf{Exp}}[\mathsf{Bad}_Q] \quad .$$

Fix any $Q \in \mathbf{Q}$. To bound the probability that Bad_Q occurs, let p be the probability that $\mathsf{xiO}.\mathsf{Eval}^{\mathcal{R}}(\widetilde{C}, z)$ queries Q for a random input $z \leftarrow \{0,1\}^n$. Then, the probability that Bad_Q occurs is exactly the probability that it is not queried in K independent evaluations with random $x_1, \dots, x_K$ and is queried in the last evaluation with a random x. This, by Fact 2.1, can be bounded by

$$\Pr_{\mathsf{Exp}'}[\mathsf{Bad}_Q] = (1-p)^K p \le 1/K \quad .$$

It follows that for $K = 100 q_o$,

$$\Pr_{\mathsf{Exp}'}[\mathsf{Bad}] \le 1/100 \quad .$$

□

Proposition 6.3. $\mathsf{xiO}^{\star}$ *preserves the security guarantee of* xiO.

Proof sketch. The new obfuscation $\widetilde{C}^{\star}$ consists exactly of an obfuscation $\widetilde{C}$ under the original scheme, plus the set $\mathcal{R}_K$, which was generated in the learning phase. However, the learning phase, simply consists of running the evaluation algorithm of the original scheme repeatedly for a polynomial number of times. Thus, any attacker against the new scheme can be perfectly simulated by an attacker for the original scheme that simulates the above handles by performing the learning process by itself. □

This completes the proof of Theorem 6.1. □

7 From FE in Oracle Models to FE in the Plain Model

We show that functional encryption (FE) schemes in the oracle models considered in the previous sections give rise to corresponding XIO schemes. We relate the succinctness required from the FE schemes to the properties required from XIO for each of our transformations. Recall that applying our transformations from the previous sections results in approximate XIO in the bilinear oracle model or in the plain model. Assuming LWE, we show a (black-box) transformation from approximate XIO to an exact FE scheme sufficient to obtain IO. We start by defining FE and then move on to the results.

7.1 Functional Encryption

We define the notions of functional encryption (FE) considered in this work. We start by defining the general syntax functional encryption. We give a unified definition for the symmetric-key and public-key settings, addressing the differences where needed.

Syntax of Functional Encryption. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ be a message domain, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ a range, and $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ a class of functions $f : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda$. A *functional encryption* (FE) scheme for $\mathcal{X}, \mathcal{Y}, \mathcal{F}$ is a tuple of polynomial-time algorithms $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ where:

- $\mathsf{Setup}(1^\lambda)$ takes as input the security parameter and outputs a master secret key MSK and master encryption key MEK.
- $\mathsf{Gen}(\mathsf{MSK}, f)$ takes as input the master secret MSK and a function $f \in \mathcal{F}$. It outputs a secret key SK_f for f.
- $\mathsf{Enc}(\mathsf{MEK}, x)$ takes as input the master encryption key MEK and a message $x \in \mathcal{X}$, and outputs a ciphertext CT.
- $\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT})$ takes as input the secret key SK_f for a function $f \in \mathcal{F}$ and a ciphertext CT, and outputs some $y \in \mathcal{Y}$, or $\perp$.

Correctness. For any message $m \in \mathcal{X}$ and function $f \in \mathcal{F}$, we require that

$$\Pr\left[\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT}) = f(m) : \begin{array}{l} (\mathsf{MSK}, \mathsf{MEK}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{SK}_f \leftarrow \mathsf{Gen}(\mathsf{MSK}, f), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MEK}, m) \end{array}\right] \geq 1 - \mathrm{negl}(\lambda) \ .$$

Remark 7.1 (Secret Key and Public Key Notation).

- In the secret-key setting we will always assume, without loss of generality, that $\mathsf{MEK} = \mathsf{MSK}$.
- In the public-key setting, we will typically denote MEK by MPK (a master public encryption key).

Definition 7.1 (K-Key Selective Security)**.** *We say that a tuple of algorithm* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a* selectively-secure*, K-key, functional encryption scheme for $\mathcal{X}, \mathcal{Y}, \mathcal{F}$, if it satisfies the following requirement, formalized by the experiment* $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, b)$ *between an adversary $\mathcal{A}$ and a challenger:*

1. *For $k \leq K(\lambda)$, the adversary submits functions $f_1, \ldots, f_k \in \mathcal{F}$ and message pairs $(x_1^0, x_1^1), \ldots, (x_m^0, x_m^1) \in \mathcal{X} \times \mathcal{X}$ to the challenger.*
2. *The challenger runs* $(\mathsf{MSK}, \mathsf{MEK}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends the adversary the ciphertexts* $\{\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MEK}, x_i^b)\}_{i \in [m]}$ *as well as the secret keys* $\{\mathsf{SK}_{f_i} \leftarrow \mathsf{Gen}(\mathsf{MSK}, f_i)\}_{i \in [k]}$*. In a public-key scheme, $\mathcal{A}$ also gets* MEK.
3. *$\mathcal{A}$ outputs a guess b' for b.*
4. *The output of the experiment is set to be the adversary's guess b' if $f_i(x_j^0) = f_i(x_j^1)$ for all $(i, j) \in [k] \times [m]$. Otherwise the output is $\perp$.*

We say that the scheme is selectively-secure if, for any polynomial-size $\mathcal{A}$, there exists a negligible function $\mu(\lambda)$, such that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PKFE}} = \left|\Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PKFE}}(1^{\lambda}, 0) = 1\right] - \Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PKFE}}(1^{\lambda}, 1) = 1\right]\right| \leq \mu(\lambda).$$

We further say that FE *is δ-selectively secure, for some concrete negligible function $\delta(\cdot)$, if for all polynomial-size distinguishers the above indistinguishability gap $\mu(\lambda)$ is smaller than $\delta(\lambda)^{\Omega(1)}$.*

Remark 7.2 (Unbounded Key). We will say that FE is *unbounded-key* if it is K-key for $K(\lambda) = \lambda^{\omega(1)}$. That is, the scheme is secure for adversaries requesting an arbitrary polynomial number of functional keys. This is sometimes also referred to as *unbounded collusion*.

Succinctness. We now consider a property of functional encryption schemes known as weak succinctness (or compactness). In general (non-succinct) FE for a function class $\mathcal{F} = \{\mathcal{F}_\lambda\}$, the time complexity of encryption may depend arbitrarily on the size of the circuit representing the function from $\mathcal{F}$. Roughly speaking, in succinct functional encryption the time to encrypt should be independent of the function class, whereas in weakly succinct functional encryption, we allow some mild dependence on the complexity of functions.

To formally capture succinctness, we consider functional encryption for collections of classes $\boldsymbol{\mathcal{F}} = \{\mathcal{F} = \{\mathcal{F}_\lambda\}\}$, such as $\mathbf{P/poly}$ or $\mathbf{NC}^1$, and address how the specific complexity of each class $\mathcal{F} \in \boldsymbol{\mathcal{F}}$, may affect encryption complexity.

Definition 7.2 (Functional Encryption for a Collection of Functions)**.** *In a functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for a collection of functions classes $\boldsymbol{\mathcal{F}} = \{\mathcal{F} = \{\mathcal{F}_\lambda\}\}$, the setup algorithm* $\mathsf{Setup}(1^\lambda, n, s)$ *gets, in addition the the security parameter 1^λ, two parameters $(n(\lambda), s(\lambda))$ representing bounds on input length and circuit size.*

For any class $\mathcal{F} = \{\mathcal{F}_\lambda\} \in \boldsymbol{\mathcal{F}}$ with maximum input length $n(\lambda)$ and maximum circuit size $s(\lambda)$, $\mathsf{FE}_{\mathcal{F}} = (\mathsf{Setup}(\cdot, n(\cdot), s(\cdot)), \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has the correctness security guarantees as defined above for general functional encryption schemes.*

Definition 7.3 (Weakly Succinct Functional Encryption)**.** *A functional encryption scheme* $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *for a collection of function classes $\boldsymbol{\mathcal{F}} = \{\mathcal{F} = \{\mathcal{F}_\lambda\}\}$ is* **weakly succinct** *if there exists a constant $\gamma < 1$, and a fixed polynomial* $\mathrm{poly}(\cdot)$*, depending only on $\boldsymbol{\mathcal{F}}$ (but not on any specific $\mathcal{F} \in \boldsymbol{\mathcal{F}}$), such that for any class $\mathcal{F} = \{\mathcal{F}_\lambda\} \in \boldsymbol{\mathcal{F}}$ with maximum input length $n(\lambda)$ and maximum circuit size $s(\lambda)$, in* $\mathsf{FE}_{\mathcal{F}}$*, the size of the encryption circuit for messages of size n is bounded by $s^\gamma \cdot \mathrm{poly}(n, \lambda)$. We call γ the compression factor. We say that the scheme is* **fully succinct** *if $\gamma = 0$. We say the the scheme is only* **weakly ciphertext succinct** *if the size limitation is only on the size of ciphertexts, and not the size of the encryption circuit.*

Functional Encryption in Oracle Models. We say that a functional encryption scheme FE is constructed relative to an oracle $\mathcal{O}$. If the corresponding algorithms, as well as the adversary in the security game, may access the oracle $\mathcal{O}$.

7.2 A Black-Box Construction of XIO from FE

In this section, we recall two constructions of XIO from secret-key FE shown in [BNPW16]. A crucial feature of these constructions is that they are *fully black-box* [RTV04] — they use the underlying FE scheme as a black-box (obliviously of the implementation of its algorithms) and their security reductions use the adversary as a black-box. In particular, these constructions relativize — starting from an FE scheme relative to any given oracle $\mathcal{O}$, we get XIO relative to the same oracle $\mathcal{O}$. We observe that the construction has the additional features required for the transformations described in the previous sections.

Construction 1: XIO From Unbounded-Key FE. Let $\mathsf{FE} = (\mathsf{Setup}^{\mathcal{O}}, \mathsf{Gen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$ be an unbounded-key, secret-key, functional encryption scheme for **P/poly**, relative to an oracle $\mathcal{O}$. In what follows, given a circuit C, we identify its input space $\{0,1\}^n$ with $[N] = \{1, \ldots, N\}$, where $N := 2^n$. Let C_y be a circuit that given as input x, returns $C(x, y)$. Also, let U_x be the universal circuit that given a circuit D as input, returns $D(x)$.

We construct an XIO scheme $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{O}}, \mathsf{xiO.Eval}^{\mathcal{O}})$, relative to the oracle $\mathcal{O}$, as follows.

The Obfuscator $\mathsf{xiO.Obf}^{\mathcal{O}}(C, 1^\lambda)$**:**

- Generates:
  - $\mathsf{MSK} \leftarrow \mathsf{Setup}^{\mathcal{O}}(1^\lambda)$,
  - $\mathsf{SK}_{U_x} \leftarrow \mathsf{Gen}^{\mathcal{O}}(\mathsf{MSK}, U_x)$ for all $x \in [N^{1/2}]$,
  - $\mathsf{CT}_y \leftarrow \mathsf{Enc}^{\mathcal{O}}(\mathsf{MSK}, C_y)$ for all $y \in [N^{1/2}]$.
- Outputs:
  - $\widetilde{C} = \left(\{\widetilde{C}_x = \mathsf{SK}_{U_x}\}_{x \in [N^{1/2}]}, \{\widetilde{C}_y = \mathsf{CT}_y\}_{y \in [N^{1/2}]}\right)$.

The Evaluator $\mathsf{xiO.Eval}^{\mathcal{O}}(\widetilde{C}, (x, y))$**:**

- Output $\mathsf{Dec}^{\mathcal{O}}(\mathsf{SK}_{U_x}, \mathsf{CT}_y)$.

Construction 2: XIO From Weakly Succinct Single-Key FE. Let $\mathsf{FE} = (\mathsf{Setup}^{\mathcal{O}}, \mathsf{Gen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$ be a single-key, secret-key, functional encryption scheme for **P/poly**, relative to an oracle $\mathcal{O}$. Assume it is weakly succinct with compression factor $\gamma = 1 - \Omega(1)$.

As above, we identify the input space $\{0,1\}^n$ with $[N] = \{1, \ldots, N\}$ and denote by C_y the circuit that given as input x, returns $C(x, y)$. Also, for a set $X = \{x\}$, let U_X be a universal circuit that given a circuit D as input, returns $(D(x) : x \in X)$.

We construct an XIO scheme $\mathsf{xiO} = (\mathsf{xiO.Obf}^{\mathcal{O}}, \mathsf{xiO.Eval}^{\mathcal{O}})$, relative to the oracle $\mathcal{O}$, as follows. Let $\alpha \in (0, 1)$ be a parameter (which we will later explain how to set).

The Obfuscator $\mathsf{xiO.Obf}^{\mathcal{O}}(C, 1^\lambda)$**:**

- Generates:
  - $\mathsf{MSK} \leftarrow \mathsf{Setup}^{\mathcal{O}}(1^\lambda)$,
  - $\mathsf{SK}_{U_X} \leftarrow \mathsf{Gen}^{\mathcal{O}}(\mathsf{MSK}, U_X)$ for $X = [N^\alpha]$,
  - $\mathsf{CT}_y \leftarrow \mathsf{Enc}^{\mathcal{O}}(\mathsf{MSK}, C_y)$ for all $y \in [N^{1-\alpha}]$.
- Outputs:
  - $\widetilde{C} = \left(\widetilde{C}_X = \mathsf{SK}_{U_X}, \{\widetilde{C}_y = \mathsf{CT}_y\}_{y \in [N^{1-\alpha}]}\right)$.

The Evaluator $\mathsf{xiO.Eval}^{\mathcal{O}}(\widetilde{C},(x,y))$**:**

- Compute $\mathsf{Dec}^{\mathcal{O}}(\mathsf{SK}_{U_X},\mathsf{CT}_y) = (z(x',y) : x' \in X)$.
- Output $z(x,y)$.

In [BNPW16], it is proven that the above two constructions satisfy the indistinguishability requirement of XIO and are correct. To be exact, the constructions there are not formulated in an oracle model, but are fully black-box. The constructions themselves use the FE algorithms in a black-box way (as can be seen above), and the security reduction is black-box in the adversary and the FE scheme.

Theorem 7.1 (follows from [BNPW16, Theorem 3.1])**.** *Constructions 1 and 2 satisfy XIO indistinguishability relative to any polynomial-size adversary* $\mathcal{A}^{\mathcal{O}}$.

Succinctness. The two constructions described above are incomparable. On one hand, the first one has better succinctness properties sufficient for all of our transformations. The second construction suffices for our first transformation from degree-d oracles to degree-2 oracles, but not for the other transformations in Sections 5,6 that completely remove the oracle. On the other hand, the first construction requires a stronger succinctness property from the underlying FE: it should be unbounded-key, which is known to imply single-key (fully-succinct) schemes through a fully black-box construction [AJS15, BV15].

We now analyze the succinctness properties of these constructions as well as how they satisfy the product form requirement needed in Section 4.

Succinctness of Construction 1. This construction is in product form with respect to the product collection

$$\mathcal{X}_n = \left\{\{x\} \;\middle|\; x \in [N^{1/2}]\right\}, \qquad \mathcal{Y}_n = \left\{\{y\} \;\middle|\; y \in [N^{1/2}]\right\} \ .$$

The time to obfuscate each piece corresponding to x is the time to derive a key $\mathsf{Gen}^{\mathcal{O}}(\mathsf{MSK},U_x)$. The time to obfuscate each piece corresponding to y is the time to encrypt $\mathsf{Enc}^{\mathcal{O}}(\mathsf{MSK},C_y)$.

Overall, for some fixed polynomial poly,

$$q_o^{\mathcal{X}}(C,\lambda) \leq \mathrm{poly}(|C|,\lambda), \qquad q_o^{\mathcal{Y}}(C,\lambda) \leq \mathrm{poly}(|C|,\lambda) \ ,$$

and overall

$$q_o(C,\lambda) = |\mathcal{X}_n|\cdot q_o^{\mathcal{X}}(C,\lambda) + |\mathcal{Y}_n|\cdot q_o^{\mathcal{Y}}(C,\lambda) \leq N^{1/2}\cdot \mathrm{poly}(|C|,\lambda) \ .$$

Evaluation corresponds to functional decryption $\mathsf{Dec}^{\mathcal{O}}(\mathsf{SK}_{U_x},\mathsf{CT}_y)$, and thus for some fixed polynomial poly,

$$q_e(C,\lambda) = \mathrm{poly}(|C|,\lambda) \ .$$

Corollary 7.1. *For the circuit class collection* ***P/poly****,*

1. *Unbounded-key FE relative to a degree-*d *oracle* $\mathcal{M}^d$*, for* $d \geq 2$*, implies approximate XIO relative to the symmetric bilinear oracle* $\mathcal{B}^2$*.*
2. *Unbounded-key FE relative to a degree-*1 *oracle* $\mathcal{M}^1$*, implies approximate XIO in the plain model.*
3. *Unbounded-key FE relative to a random oracle* $\mathcal{R}$*, implies approximate XIO in the plain model.*

Proof. For the first item, we note that

$$\begin{aligned}
&|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}} \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right)\right)^d && +|\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}} \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right)\right)^d && = \\
&N^{1/2} \cdot \mathrm{poly}(|C|, \lambda) && +N^{1/2} \cdot \mathrm{poly}(|C|, \lambda) && = 2^{n/2} \cdot \mathrm{poly}(|C|, \lambda) \ ,
\end{aligned}$$

satisfying the conditions for the transformation from Section 4 (Theorem 4.1).

For the second and third items, we have that

$$q_o \cdot q_e = N^{1/2} \cdot \mathrm{poly}(|C|, \lambda) = 2^{n/2} \cdot \mathrm{poly}(|C|, \lambda) \ ,$$

satisfying the conditions for the transformations from Section 5 and Section 6 (Theorem 5.1 ad Theorem 6.1). □

Remark 7.3. We note that above the requirement that the size of the encryption circuit is *completely independent* of the number of keys, can be relaxed. For instance, for Theorems 5.1 and 6.1, it suffices that the size of the encryption circuit is bounded by $N^{\frac{1}{4}-\Omega(1)} \cdot \mathrm{poly}(\lambda)$, where N is the number of released keys.

Succinctness of Construction 2. This construction is in product form with respect to the product collection

$$\mathcal{X}_n = \{X = [N^\alpha]\} , \qquad \mathcal{Y}_n = \left\{\{y\} \mid y \in [N^{1-\alpha}]\right\} \ .$$

The time to obfuscate the piece corresponding to X is the time to derive a key $\mathsf{Gen}^{\mathcal{O}}(\mathsf{MSK}, U_X)$. Thus for some fixed poly,

$$q_o^{\mathcal{X}}(C, \lambda) \leq \mathrm{poly}(|U_X|, \lambda) = \mathrm{poly}(N^\alpha, |C|, \lambda) = N^{\alpha c} \cdot \mathrm{poly}(|C|, \lambda) \ ,$$

for some constant c that depends only on the Gen algorithm.

The time to obfuscate each piece corresponding to y is the time to encrypt $\mathsf{Enc}^{\mathcal{O}}(\mathsf{MSK}, C_y)$.

$$q_o^{\mathcal{Y}}(C, \lambda) \leq |U_X|^\gamma \cdot \mathrm{poly}(|C|, \lambda) = N^{\alpha\gamma} \cdot \mathrm{poly}(|C|, \lambda) \ ,$$

where γ is the compression factor of the FE scheme.

Corollary 7.2. *Single-key FE for* $\boldsymbol{P}/\boldsymbol{poly}$*, relative to a degree-$d$ oracle $\mathcal{M}^d$, with weak succinctness and compression factor $\gamma < 1/d$ implies approximate XIO relative to the symmetric bilinear oracle $\mathcal{B}^2$.*

Proof. We note that

$$\begin{aligned}
&|\mathcal{X}_n| \cdot \left(q_o^{\mathcal{X}} \cdot \min\left(q_o^{\mathcal{X}}, |\mathcal{Y}_n| \cdot \log q_o^{\mathcal{X}}\right)\right)^d && +|\mathcal{Y}_n| \cdot \left(q_o^{\mathcal{Y}} \cdot \min\left(q_o^{\mathcal{Y}}, |\mathcal{X}_n| \cdot \log q_o^{\mathcal{Y}}\right)\right)^d && \leq \\
&1 \cdot (N^{\alpha c} \cdot N^{\alpha c})^d \cdot \mathrm{poly}(|C|, \lambda) && +N^{1-\alpha} \cdot (N^{\alpha\gamma} \cdot 1)^d \cdot \mathrm{poly}(|C|, \lambda) && = 2^{n \cdot (1-\Omega(1))} \cdot \mathrm{poly}(|C|, \lambda) \ ,
\end{aligned}$$

for any $\gamma < 1/d$, and setting, in construction 2, $\alpha < 1/2cd$.

This satisfies the conditions for the transformation from Section 4 (Theorem 4.1). □

We note that the transformations from Sections 5,6 cannot be applied here as $q_o \geq N^{1-\alpha}$ and $q_e \geq N^\alpha$, implying that $q_o q_e \geq 1$, which does not satisfy the basic requirements for these transformations.

7.3 From (Approximate) XIO and LWE to FE

We show how to use approximate XIO to construct 1-key weakly succinct FE for **P**/**poly**, assuming LWE.

Remark 7.4 (Modulus-to-Noise). We rely (as a black-box) on previous results [GKP+13, LPST16a], which assume LWE. These rely on the LWE assumptions with a *quasi-polynomial modulus-to-noise ratio*. To be explicit, we will denote this assumption by $\mathsf{LWE}_{\mathsf{qpr}}$ (for "quasi-polynomial ratio").

Theorem 7.2. *Assuming* $\mathsf{LWE}_{\mathsf{qpr}}$ *and the existence of an approximate XIO scheme for* $\mathbf{P}^{\log}/\mathbf{poly}$*, there exists a single-key weakly-succinct FE scheme for* $\mathbf{P}/\mathbf{poly}$*.*

We first describe our ideas at a high-level and then provide the formal transformation in Section 7.3.1.

A Failed Attempt. Lin, Pass, Seth and Telang [LPST16a] showed a transformation from correct XIO for $\mathbf{P}^{\log}/\mathbf{poly}$ to IO for **P**/**poly**, assuming LWE.[7] Previously, Bitansky and Vaikuntanathan [BV16] showed how to make any approximately correct IO correct (assuming, say, LWE). Thus, to prove the above theorem, a natural idea is to combine the two — amplify the correctness of approximate XIO to obtain correct XIO by [BV16], and then invoke the transformation of [LPST16a]. This approach turns out to completely fail. Indeed, the [BV16] transformation only works for classes of circuits that are expressive enough; in particular, it relies on the ability of circuits in the class to process encrypted inputs, which must inherently be of super-logarithmic length in the security parameter. However, XIO for such circuit classes, which lie outside of $\mathbf{P}^{\log}/\mathbf{poly}$, is inefficient (see Remark 2.1).

Instead, we show how to modify the transformation of [LPST16a], based on error-correcting codes, so that it works directly with approximate XIO. Below, we briefly review the [LPST16a] transformation and describe our key ideas.

Review of the [LPST16a] Transformation. Goldwassar et al. [GKP+13] constructed, from LWE with quasi-polynomial modulus-to-noise ratio, a fully succinct, public-key, single-key, FE scheme for *Boolean* $\mathbf{NC}^1$ circuits; namely, the encryption circuit of their scheme has size $\mathrm{poly}(n, \lambda)$, where n is the message length. (They, in fact, show a result for any polynomial-size circuit class, where the encryption circuit and the modulus-to-noise ratio grow with the depth. For our contexts, we can just focus on $\mathbf{NC}^1$.)

Theorem 7.3 (Fully Succinct FE scheme for Boolean $\mathbf{NC}^1$ [GKP+13])**.** *Assume* $\mathsf{LWE}_{\mathsf{qpr}}$*. There exists a fully succinct, public-key, one-key, FE scheme for any collection* $\mathcal{C} \subseteq \mathbf{NC}^1$ *of Boolean circuit classes.*

Starting from such an FE scheme bFE for Boolean circuits, the first observation in [LPST16a] is as follows: To construct an FE scheme FE, for any (possibly non-Boolean) circuit C, one can use bFE to issue a key for the corresponding Boolean circuit B that produces *one output bit at a time*, that is, $B(m, i) = (C(m))_i$. Then, to enable evaluating the circuit C, it suffices to publish a list of bFE ciphertexts encrypting all pairs (m, i). This, however, leads to a scheme with encryption time linear in the length of the output (as it needs to produce a ciphertext for every output bit), and is not weakly succinct. The key idea in [LPST16a] is using XIO to generate the list of encrypted pairs (m, i). Namely, obfuscate a circuit that given as input i, outputs the encryption of (m, i), where randomness is derived with a pseudorandom function. Since XIO achieves "sublinear compression", the resulting FE scheme is now weakly succinct for all of $\mathbf{NC}^1$, including circuits with non-Boolean output.

Our Approach. The basic idea behind replacing XIO with approximate XIO is to use good error-correcting codes to allow recovering the output of a given function even if some of the encryptions (m, i) are faulty.

[7] The LWE assumption was later weakened to the existence of public key encryption by [BNPW16], but only for sufficiently-compressing XIO.

Specifically, we make the following modification to the transformation of [LPST16a]. Instead of deriving a key for the Boolean function $B(m,i) = (C(m))_i$, which computes the i-th bit of the circuit's output, we consider the function $B^\star(m,i) = (\mathsf{ECC}(C(m)))_i$ that outputs the i-th bit of an error-corrected version of this output. As before, we use XIO to to generate the list of encryptions (m,i), only that now, with approximate XIO, some of these encryptions may be faulty. Nevertheless, we can still recover $(\mathsf{ECC}(C(m)))_i$ for a large enough fraction of indices i, and can thus correct, and obtain $C(m)$. By using codes with constant rate, and a linear-size constant-depth encoding circuit, we can show that this transformation achieves the required compression.

Reformulating Approximate XIO. Before formally describing our transformation, we first show the following claim that gives a slightly amplified notion of correctness for approximate XIO, which would be easier for us to work with. In the definition of approximate XIO (Definition 2.2), we allowed a constant error not only over the inputs, but also over the randomness of the obfuscator, in particular, there could be a constant fraction of coins, for which there is no correctness at all. In the amplified version considered next, it is guaranteed that, with overwhelming probability over the coins of the obfuscator, there is a small (constant) fraction of erroneous inputs.

Claim 7.1. *Approximate correctness of XIO can be amplified to the following correctness guarantee:*

$$\Pr_{\mathsf{xiO.Obf}}\left[\Pr_{x\leftarrow\{0,1\}^n}[\mathsf{xiO.Eval}(\widetilde{C},x) = C(x)] \geq 0.9 : \widetilde{C} \leftarrow \mathsf{xiO.Obf}(C)\right] \geq 1 - \mathrm{negl}(\lambda) \ .$$

Proof. In what follows, let $\mathsf{xiO} = (\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ be an approximate XIO scheme for a collection of circuit classes $\mathcal{C} \subseteq \mathbf{P}^{\log}/\mathbf{poly}$. We describe a new scheme $\mathsf{xiO}^\star = (\mathsf{xiO}^\star.\mathsf{Obf}, \mathsf{xiO}^\star.\mathsf{Eval})$ that satisfies the correctenss requirement given by the claim. The new obfuscator $\mathsf{xiO}^\star.\mathsf{Obf}(C, 1^\lambda)$ simply outputs λ indpendnent obfuscations $\widetilde{C}_1, \dots, \widetilde{C}_\lambda \leftarrow \mathsf{xiO.Obf}(C, 1^\lambda)$. The new evaluator $\mathsf{xiO}^\star.\mathsf{Eval}$, for input x, simply evaluates each of the λ obfuscations on x, and outputs the majority result.

First, note that the new obfuscator satisfies the XIO indistinguishability requirement, by a straightforward hybrid argument. We now show that it satisfies the correctness requirement given by the claim. Fix any circuit C, and consider the set of inputs for which the obfuscator $\mathsf{xiO.Obf}$ is correct with high probability

$$S_C = \left\{ x \;\middle|\; \Pr_{\mathsf{xiO.Obf}}[\mathsf{xiO.Eval}(\widetilde{C},x) = C(x) : \widetilde{C} \leftarrow \mathsf{xiO.Obf}(C)] \geq 0.9 \right\} \ .$$

Then since

$$\Pr_{x,\mathsf{xiO.Obf}}[\mathsf{xiO.Eval}(\widetilde{C},x) = C(x) : \widetilde{C} \leftarrow \mathsf{xiO.Obf}(C)] \geq 0.99 \ ,$$

it follows by averaging that $\Pr_x[x \in S_c] \geq 0.9$.

To conclude the claim, we show that

$$\Pr_{\mathsf{xiO}^\star.\mathsf{Obf}}\left[\forall x \in S_C : \mathsf{xiO}^\star.\mathsf{Eval}(\widetilde{C}^\star, x) = C(x) \;\middle|\; \widetilde{C}^\star \leftarrow \mathsf{xiO}^\star.\mathsf{Obf}(C)\right] \geq 1 - \mathrm{negl}(\lambda) \ .$$

Indeed, for any $x \in S_C$ the majority of $\mathsf{xiO.Eval}(\widetilde{C}_i, x)$ equals $C(x)$, except with probability $2^{-\Omega(\lambda)}$, over the coins of $\mathsf{xiO}^\star.\mathsf{Obf}$. The required bound follows by taking a union bound over all $x \in S_C$, and the fact that $|S_C| \leq 2^n \leq 2^{O(\log\lambda)}$. □

7.3.1 The Transformation

Our construction of 1-key FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for $\mathbf{P}/\mathbf{poly}$ relies on the following tools:

- An approximate XIO scheme $\mathsf{xiO} = (\mathsf{xiO.Obf}, \mathsf{xiO.Eval})$ for $\mathbf{P}^{\log}/\mathbf{poly}$ (with correctness as given by Claim 7.1).
- A fully succinct, public-key, 1-key, FE scheme $\mathsf{bFE} = (\mathsf{bSetup}, \mathsf{bGen}, \mathsf{bEnc}, \mathsf{bDec})$ for Boolean $\mathbf{NC}^1$ (Theorem 7.3).
- An error-correcting code $(\mathsf{ECC}, \mathsf{Decode})$ with constant rate that tolerates 0.1-error rate, and has a linear-size, low-depth, encoding algorithm $\mathsf{ECC} \in \mathbf{NC}^1$ [Spi96].[8]
- A puncturable PRF scheme $\mathcal{PPRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$ (Definition 2.3).

In what follows, $n = n(\lambda)$ and $s = s(\lambda)$ denote bounds on the input length and size of circuits; s is in particular also a bound on the number of output bits. Also, given that we are dealing with public-key FE schemes, we shall denote the encryption key by MPK.

Setup $\mathsf{Setup}(1^\lambda, n, s)$**:** Samples $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{bSetup}(1^\lambda, n', s')$. Here $n' \leq n + O(\log s), s' \leq O(s)$; we explain exactly how these parameters are chosen below.

Key Generation $\mathsf{Gen}(\mathsf{MSK}, C)$**:** Construct the following Boolean circuit.

- B is a Boolean circuit that computes the ECC encoding of the outputs of C, one bit at a time. That is, for every $m \in \{0,1\}^n$ and index $i \in \{0,1\}^{\log \ell}$, $B(m, i) = \mathsf{ECC}(C(m))_i$, where i is an index in the ECC encoding of C's outputs and is upper bounded by $\ell = |\mathsf{ECC}(C(m))| = O(s)$.

Sample $\mathsf{bSK}_B \leftarrow \mathsf{bGen}(\mathsf{MSK}, B)$ and output $\mathsf{SK}_C = \mathsf{bSK}_B$.

Encryption $\mathsf{Enc}(\mathsf{MPK}, m)$**:**

- Sample a PPRF key $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$.
- Use XIO to obfuscate the circuit $E = E[\mathsf{MPK}, m, K]$ that receives as input an index $i \in \{0,1\}^{\log \ell}$, and outputs the ciphertext $\mathsf{bCT} = \mathsf{bEnc}(\mathsf{MPK}, (m, i); \mathsf{PRF.Eval}_K(i))$ (Figure 1). Obtain obfuscated circuit $\widetilde{E} \leftarrow \mathsf{xiO.Obf}(E, 1^\lambda)$.
- Output the ciphertext $\mathsf{CT} = \widetilde{E}$.

Decryption $\mathsf{Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** Parse CT as an obfuscated program $\widetilde{E}$ and do the following for every $i \in [\ell]$,

- Compute $\mathsf{bCT}_i = \mathsf{xiO.Eval}(\widetilde{E}, i)$.
- Decrypt $c_i = \mathsf{bDec}(\mathsf{bSK}_B, \mathsf{bCT}_i)$.
- Decode $c = c_1 \dots c_\ell$ to obtain $y = \mathsf{Decode}(c)$, and output y.

Theorem 7.4. *The above scheme* FE *is a one-key, public-key, weakly ciphertext-succinct, FE scheme for any circuit collection* $\boldsymbol{\mathcal{C}} \subseteq \boldsymbol{NC}^1$.

[8] The requirement for a linear-size encoding circuit is not essential, but is achieved by [Spi96], and will simplify parameters.

Circuit $E[\mathsf{MPK}, m, K](i)$

Constants: A public encryption key MPK of bFE, message $m \in \{0,1\}^n$, a key K of PPRF.

Input: An index $i \in [\ell]$.

Procedure:

1. Evaluate PPRF on input index i, $R_i = \mathsf{PRF.Eval}_K(i)$;
2. Encrypt pair (m, i), $\mathsf{bCT}_i = \mathsf{bEnc}(\mathsf{MPK}, (m, i)\ ;\ R_i)$.

Output: Output bCT_i.

Padding: The circuit E is padded to be of the same size as circuit $E^\star$ in Figure 2.

Figure 1: Circuit E used in the construction of the 1-key succinct FE scheme FE

In [LPST16b], they show a transformation, assuming LWE, from ciphertext succinctness to weak succinctness (where the encryption circuit size is bounded, rather than only ciphertext size) for $\mathbf{NC}^1$ (with the same compression factor). In [ABSV15], they show a transformation from any scheme for $\mathbf{NC}^1$ to a scheme for $\mathbf{P}/\mathbf{poly}$, the transformation preserves weak succinctness with the same compression factor.

Corollary 7.3 (from [ABSV15, LPST16b])**.** *Assuming* $\mathsf{LWE}_{\mathsf{qpr}}$ *and approximate XIO for* $\mathbf{P}/\mathbf{poly}$*, there is a one-key, public-key, weakly succinct, FE scheme for any circuit collection* $\mathcal{C} \subseteq \mathbf{P}/\mathbf{poly}$.

Proof of Theorem 7.4. First, we note that for any collection $\mathcal{C}$ in $\mathbf{NC}^1$, considering the circuits B for circuits in $\mathcal{C}$ results in a collection $\mathcal{B}$ also in $\mathbf{NC}^1$, as supported by the Boolean FE scheme bFE. Indeed, for any circuit $C \in \mathcal{C}$, B consists of executing C, error-correcting the output, and then choosing an output bit. Since we are using a linear log-depth error-correcting code, the circuit size is $O(|C|)$ and its depth is $O(\log n)$.

We next prove correctness, succinctness, and security.

Proposition 7.1. *The scheme* FE *is correct.*

Proof. The correctness follows directly from the approximate correctness of xiO, correctness of the Boolean FE scheme bFE, and the correction guarantee of the error-correcting code ECC. Indeed, by the approximate correctness of XIO, except with negligible probability, for each ciphertext $\mathsf{CT} = \widetilde{E}$ generated by FE, it is the case that $\mathsf{xiO.Eval}(\widetilde{E}, i)$ outputs the correct cipertext CT_i, under the Boolean bFE, for at lest 0.9 fraction of i's. By the correctness of bFE, decryption results in the correct value $(\mathsf{ECC}(C(m)))_i$, for each correct ciphertext, and decoding will recover the correct value $C(m)$. □

Proposition 7.2. *The scheme* FE *is ciphertext succinct.*

Proof. By construction of FE, the size of any ciphertext CT is exactly the size of the obfuscated circuit $\widetilde{E}$, which we now bound. Let γ be the compression factor of the XIO scheme. Then, by the XIO guarantee:

$$|\widetilde{E}| \le \ell^\gamma \cdot \mathrm{poly}(|\widetilde{E}|, \lambda) \le s^\gamma \cdot \mathrm{poly}(n, \lambda) \ .$$

Indeed, recall that ℓ is the bound on the size of codewords $\mathsf{ECC}(C(m))$, where $|C(m)| \le s$, and since we are using linear error-correcting codes, $\ell \le O(s)$. The size $|\widetilde{E}|$ is determined by computing a puncturable

PRF and an encryption of a message of size at most $n + \log \ell$, and padded by at most $\mathrm{poly}(n, \lambda)$ (Figure 2). Thus $|\widetilde{E}| \le \mathrm{poly}(n, \lambda)$. We conclude that the scheme is ciphertext succinct.

$\square$

Proposition 7.3. *The scheme* FE *is secure.*

Proof. We show that for any polynomial-size adversary $\mathcal{A}$, the experiments $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ and $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1)$ are indistinguishable. Since we are in the public-key setting, it suffices to focus on the case that the adversary requests single ciphertext. Specifically, in $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, b)$, $\mathcal{A}$ chooses a challenge circuit C and a pair of challenge inputs (m_0, m_1) such that $C(m_0) = C(m_1)$, and receives a secret key SK_C for C and a ciphertext CT_b encrypting m_b. By construction of FE, the challenge ciphertext CT_b contains an obfuscation of the circuit $E_b = E[\mathsf{MPK}, m_b, K]$.

We show the indistinguishability of $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ and $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1)$ using the following sequence of hybrids $\{H^\alpha_{i^*}\}$ for $\alpha = 0, \cdots, 3$ and $i^* \in [\ell + 1]$.

Hybrid $H^0_{i^*}$ proceeds identically to $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ except that the challenge ciphertext contains an obfuscation of the following circuit $E^\star$ as described in Figure 2:

$$E^\star_{i^*,0} = E^\star[\mathsf{MPK}, \underline{K\{i^*\}, i^*, m_0, m_1, \mathsf{bCT}^{(m_0,i^*)}}] ,$$
$$\text{where } \mathsf{bCT}^{(m_0,i^*)} = \mathsf{bEnc}(\mathsf{MPK}, \underline{(m_0, i^*)\ ;\ \mathsf{PRF.Eval}_K(i^*)}).$$

Circuit $E^\star[\mathsf{MPK}, K, \underline{i^*, m_<, m_>, \mathsf{bCT}^*}](i)$

Constants: A public encryption key MPK of bFE, a key K of PPRF, a threshold $i^* \in \{0, \cdots, \ell + 1\}$, strings $m_<, m_> \in \{0, 1\}^n$, and a ciphertext bCT^* of bFE.

Input: An index $i \in [\ell]$.

Procedure:

1. Evaluate PPRF on input index i, $R_i = \mathsf{PRF.Eval}_K(i)$;
2. If $\underline{i < i^*}$, encrypt pair $(m_<, i)$, $\mathsf{bCT}_i = \mathsf{bEnc}(\mathsf{MPK}, (\underline{m_<}, i)\ ;\ R_i)$.
3. If $\underline{i = i^*}$, set $\underline{\mathsf{bCT}_i = \mathsf{bCT}^*}$.
4. If $\underline{i > i^*}$, encrypt pair $(m_>, i)$, $\mathsf{bCT}_i = \mathsf{bEnc}(\mathsf{MPK}, (\underline{m_>}, i)\ ;\ R_i)$.

Output: Output the ciphertext bCT_i.

Figure 2: Circuit $E^\star$ used in the security analysis of the FE scheme FE

We claim that $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0) \approx_c H^0_1$. This is because the only difference between these two games is that in the former circuit E_0 is obfuscated during encryption, whereas in the latter $E^\star_{1,0}$ is obfuscated. Since $E_0 \equiv E^\star_{1,0}$ (namely, the two circuits have the same truth table and same size), by the security of XIO, their obfuscations are indistinguishable. Thus, so are the two hybrids.

Similarly, $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1) \approx_c H^0_{\ell+1}$, by the same argument and the fact that $E_0 \equiv E^{\ell+1,0}$.

Hybrid $H^1_{i^*}$ proceeds identically to $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ except that the challenge ciphertext contains an obfuscation of the following circuit

$$E^\star_{i^*,1} = E[\mathsf{MPK}, K\{i^*\}, i^*, m_0, m_1, \underline{\mathsf{bCT}_r^{(m_0,i^*)}}] \,,$$
$$\text{where } \mathsf{bCT}_r^{(m_0,i^*)} = \mathsf{bEnc}(\mathsf{MPK}, (m_0, i)\ ;\ \underline{r}) \text{ with random } r.$$

We claim that $H^0_{i^*} \approx_c H^1_{i^*}$. This is because by the security of punctured PRF, $\mathsf{PRF.Eval}_K(i^*)$ (in H_{i^*}) is pseudorandom, given the punctured PRF key $K\{i^*\}$.

Hybrid $H^2_{i^*}$ proceeds identically to $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ except that the challenge ciphertext contains an obfuscation of the following circuit

$$E^\star_{i^*,2} = E[\mathsf{MPK}, K\{i^*\}, i^*, m_0, m_1, \underline{\mathsf{bCT}_r^{(m_1,i^*)}}] \,,$$
$$\text{where } \mathsf{bCT}_r^{(m_1,i^*)} = \mathsf{bEnc}(\mathsf{MPK}, \underline{(m_1, i^*)}\ ;\ r) \text{ with random } r.$$

We have $H^1_{i^*} \approx_c H^2_{i^*}$, since by the security of scheme bFE, ciphertexts $\mathsf{bCT}_r^{(m_0,i^*)}$ and $\mathsf{bCT}_r^{(m_1,i^*)}$ are indistinguishable, given the public encryption key MPK.

Hybrid $H^3_{i^*}$ proceeds identically to $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$, except that the challenge ciphertext contains an obfuscation of the following circuit

$$E^\star_{i^*,3} = E[\mathsf{MPK}, K\{i^*\}, i^*, m_0, m_1, \underline{\mathsf{bCT}^{(m_1,i^*)}}] \,,$$
$$\text{where } \mathsf{bCT}^{(m_1,i^*)} = \mathsf{bEnc}(\mathsf{MEK}, (m_1, i^*)\ ;\ \underline{\mathsf{PRF.Eval}K(i^*)}).$$

We have $H^2_{i^*} \approx_c H^3_{i^*}$, since by the security of punctured PRF, $\mathsf{PRF.Eval}K(i^*)$ (in $H^3_{i^*}$) is pseudorandom, given the punctured PRF key $K\{i^*\}$.

Furthermore, we claim that $H^3_{i^*} \approx H^0_{i^*+1}$. This is because $E^\star_{i^*,3} \equiv E^\star_{i^*+1,0}$, and by the security of XIO, their obfuscation are indistinguishable. Thus, so are the two hybrids.

Therefore, $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 0)$ and $\mathsf{Expt}^{\mathsf{FE}}_{\mathcal{A}}(1^\lambda, 1)$ are indistinguishable, which concludes the proof of the proposition. □

This concludes the proof of Theorem 7.4. □

7.4 Putting it All Together

We now stitch the different pieces together. Starting with a single-key weakly-succinct FE scheme for **P/poly** relative to a degree-d decomposable oracle, with compression factor $\gamma < 1/d$, by Corollary 7.2, we can convert it into an approximately correct XiO scheme relative to the symmetric bilinear oracle $\mathcal{B}^2$. Theorem 7.2 states that approximate XiO for $\mathbf{P}^{\log}/\mathbf{poly}$ implies single-key weakly-succinct FE for **P/poly** assuming LWE, through a fully black-box construction. Since the construction is fully black-box, the implication holds even relative to oracle $\mathcal{B}^2$. Therefore, combining Corollary 7.2 and Theorem 7.2 gives a single-key weakly-succinct FE scheme relative to $\mathcal{B}^2$.

Theorem 7.5. *Assume the hardness of* $\mathsf{LWE}_{\mathsf{qpr}}$*. Single-key FE for* $\mathbf{P}/\mathbf{poly}$ *relative to a degree-*d *decomposable oracle* $\mathcal{M}^d$ *with weak succinctness and compression factor* $\gamma < 1/d$ *implies single-key FE for* $\mathbf{P}/\mathbf{poly}$ *relative to the symmetric bilinear oracle* $\mathcal{B}^2$ *with weak succinctness. Assuming additionally an über assumption on symmetric bilinear pairing groups, the former implies single-key FE in the plain model.*

Starting with an unbounded-key FE scheme for $\mathbf{P}/\mathbf{poly}$, we can combine Corollary 7.1 and Theorem 7.2 to get the following theorem for the random-oracle model and the degree-one graded encoding model (akin to the generic-group model).[9]

Theorem 7.6. *Assume the hardness of* $\mathsf{LWE}_{\mathsf{qpr}}$*. Unbounded-key FE relative to a degree-*1 *oracle* $\mathcal{M}^1$*, or a random oracle* $\mathcal{R}$*, implies single-key FE for* $\mathbf{P}/\mathbf{poly}$ *in the plain model.*

We note that all of our reductions are *polynomial*. In particular, if the primitives we start with are subexponentiall-secure (and not just polynomially-secure), the resulting FE is also subexponentially-secure and can be used to obtain IO [AJ15, BV15].

Acknowledgements

We thank Vinod Vaikuntanathan for enlightening discussions.

References

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

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 657–677, 2015.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. *IACR Cryptology ePrint Archive*, 2015:730, 2015.

[AS16] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. *IACR Cryptology ePrint Archive*, 2016:1097, 2016.

[9] Note that unbounded-key FE is also sufficient for the previous theorem as it implies fully-succinct FE in a black-box way (see Section 7.2).

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 440–456, 2005.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.

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

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016: 7th Innovations in Theoretical Computer Science*, pages 345–356, Cambridge, MA, USA, January 14–16, 2016. Association for Computing Machinery.

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

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In *Theory of Cryptography - 14th International Conference, TCC 2016-B*, 2016.

[Boy08] Xavier Boyen. The uber-assumption family. In *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, pages 39–56, 2008.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 1480–1498, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.

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

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190, 2015.

[BV16] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation: From approximate to exact. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 67–95, 2016.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. *IACR Cryptology ePrint Archive*, 2014:930, 2014.

[BZ16] Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 176–206, 2016.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 247–266, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

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

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 456–467, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

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

[GGH13b] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 1–17, 2013.

[GGH+13c] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th*

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

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 480–511, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Unifying security notions of functional encryption. *IACR Cryptology ePrint Archive*, 2016:524, 2016.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179, 2012.

[Jou02] Antoine Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings*, pages 20–32, 2002.

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press.

[Lin16a] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 28–57. Springer, 2016.

[Lin16b] Huijia Lin. Indistinguishability obfuscation from DDH on 5-linear maps and locality-5 prgs. *IACR Cryptology ePrint Archive*, 2016:1096, 2016.

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. *IACR Cryptology ePrint Archive*, 2016:561, 2016.

[LPST16a] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 447–462, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[LPST16b] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 96–124, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016*, 2016.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany.

[MMN16] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. On the impossibility of virtual black-box obfuscation in idealized models. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 18–48, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 629–658, 2016.

[PS16] Rafael Pass and Abhi Shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 3–17, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, pages 1–20, 2004.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Information Theory*, 42(6):1723–1731, 1996.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10: 17th Conference on Computer and Communications Security*, pages 463–472, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

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