# On the Complexity of Compressing Obfuscation

Gilad Asharov* Naomi Ephraim† Ilan Komargodski‡ Rafael Pass§

**Abstract**

Indistinguishability obfuscation has become one of the most exciting cryptographic primitives due to its far reaching applications in cryptography and other fields. However, to date, obtaining a plausibly secure construction has been an illusive task, thus motivating the study of seemingly weaker primitives that imply it, with the possibility that they will be easier to construct.

In this work, we provide a systematic study of compressing obfuscation, one of the most natural and simple to describe primitives that is known to imply indistinguishability obfuscation when combined with other standard assumptions. A compressing obfuscator is roughly an indistinguishability obfuscator that outputs just a slightly compressed encoding of the truth table. This generalizes notions introduced by Lin et al. (PKC 2016) and Bitansky et al. (TCC 2016) by allowing for a broader regime of parameters.

We view compressing obfuscation as an independent cryptographic primitive and show various positive and negative results concerning its power and plausibility of existence, demonstrating significant differences from full-fledged indistinguishability obfuscation.

First, we show that as a cryptographic building block, compressing obfuscation is weak. In particular, when combined with one-way functions, it cannot be used (in a black-box way) to achieve public-key encryption, even under (sub-)exponential security assumptions. This is in sharp contrast to indistinguishability obfuscation, which together with one-way functions implies almost all cryptographic primitives.

Second, we show that to construct compressing obfuscation with perfect correctness, one only needs to assume its existence with a very weak correctness guarantee and polynomial hardness. Namely, we show a correctness amplification transformation with optimal parameters that relies only on polynomial hardness assumptions. This implies a universal construction assuming only polynomially secure compressing obfuscation with approximate correctness. In the context of indistinguishability obfuscation, we know how to achieve such a result only under sub-exponential security assumptions together with derandomization assumptions.

Lastly, we characterize the existence of compressing obfuscation with *statistical* security. We show that in some range of parameters and for some classes of circuits such an obfuscator *exists*, whereas it is unlikely to exist with better parameters or for larger classes of circuits. These positive and negative results reveal a deep connection between compressing obfuscation and various concepts in complexity theory and learning theory.

*Cornell Tech, New York, NY 10044, USA. Email: asharov@cornell.edu. Supported by a Junior Fellow award from the Simons Foundation.

†Cornell University, Ithaca, NY 14853, USA. Email: nephraim@cs.cornell.edu. Supported by an AFOSR grant FA9550-15-1-0262.

‡Cornell Tech, New York, NY 10044, USA. Email: komargodski@cornell.edu. Supported in part by a Packard Foundation Fellowship and by an AFOSR grant FA9550-15-1-0262.

§Cornell Tech, New York, NY 10044, USA. Email: rafael@cs.cornell.edu. Supported in part by NSF Award CNS-1561209, NSF Award CNS-1217821, NSF Award CNS-1704788, AFOSR Award FA9550-15-1-0262, a Microsoft Faculty Fellowship, and a Google Faculty Research Award.

# Contents

# 1 Introduction

Program obfuscation is an intriguing and powerful concept in modern cryptography. A program obfuscator is a compiler that "scrambles" programs into ones that are hard to reverse engineer, while preserving their functionality. The predominant notion that captures the above concept is *indistinguishability obfuscation*, introduced in the seminal work of Barak et al. [15], which has inspired a vibrant area of research in recent years. Informally, indistinguishability obfuscation (iO) guarantees that the obfuscations of two functionally equivalent circuits of the same size are computationally indistinguishable.

There are two main reasons why iO has become such a central primitive—its potential to exist and its power. As opposed to stronger notions of obfuscation that are known not to exist for all circuits (such as *virtual black-box* obfuscation [15]), general purpose iO might be realizable, and in fact, since the work of Garg et al. [45] many candidate constructions of iO have emerged [45, 33, 14, 6, 88, 51, 9, 93, 50]. As for its power, iO serves as a hub for an impressive number of cryptographic primitives, ranging from classical concepts such as one-way functions [70], public-key encryption [90], trapdoor permutations [20], ZAPs and non-interactive witness-indistinguishable proofs [19], to ones that are still far beyond the reach of any other assumption, such as deniable encryption [90], fully-secure multi-input functional encryption [54], and many others.

Despite immense efforts to construct iO from concrete assumptions, all currently known candidate constructions have been shown to be vulnerable to attacks [8, 13, 27, 38, 39, 40, 80, 84].[1] Another line of work shows how to construct iO from some seemingly "simpler" or "weaker" generic cryptographic primitives (together with more standard assumptions). These include primitives such as low-degree multilinear maps [73, 74, 5, 77], compact functional encryption schemes [4, 21], compact randomized encodings [76], and variants of exponentially-efficient indistinguishability obfuscation [18, 75], all of which have no known instantiations from standard assumptions.

The difficulty of constructing iO motivates the study of such seemingly weaker cryptographic primitives, with the hope that such a study could elucidate the foundations of iO. In this paper, we focus on the primitive which is arguably the simplest to define and the closest in its nature to iO: indistinguishability obfuscation with nontrivial compression, or in short, *compressing obfuscation*.

**Compressing obfuscation.** For functions $t(s,n)$ and $\ell(s,n)$, we say that an obfuscator $\mathcal{O}$ is $(t,\ell)$-compressing if, when given a circuit $C$ of size $s$ on $n$ inputs, the obfuscator $\mathcal{O}(C)$ runs in time $t(s,n)$ and has output length $\ell(s,n)$. In the case of iO, both $t$ and $\ell$ are polynomial in $s$ and $n$, but in general, we allow them to be super-polynomial, or even (sub-)exponential. This definition generalizes existing relaxations of iO (such as XiO and SXiO which we discuss below) and allows us to characterize the extent to which efficiency impacts the existence, applications, and limitations of obfuscation. Throughout this work, we mostly focus on the following two settings of parameters, which intuitively, are relaxed versions of iO that only allow obfuscating circuits with logarithmic input size:

- **XiO.** The first (and weaker) setting of parameters is that of *exponentially-efficient iO* (XiO), introduced by Lin et al. [75]. XiO allows the running time of the obfuscator to be as large as the truth table of the circuit to be obfuscated, but requires the size of the obfuscated circuit to be slightly smaller than its truth table. More formally, for a function $c$ (which denotes the compression of XiO), we say that $c$-XiO is a $(t,\ell)$-compressing obfuscator with $t(s,n) = \mathsf{poly}(2^n, s)$ and $\ell(s,n) = c(n) \cdot \mathsf{poly}(s)$. When there exists a constant $\epsilon > 0$ such that $c(n) = 2^{n(1-\epsilon)}$, we denote $c$-XiO simply by XiO. Lin et al. [75] showed that XiO for all circuits and Learning With Errors (LWE), both with sub-exponential security, imply iO.

[1] Some of the attacks apply directly to the candidate construction while some only apply to the underlying graded encoding scheme [41, 50, 42]. See Ananth et al. [2, Appendix A] for an overview.

– **SXiO.** The second (and stronger) setting of parameters is that of *strong XiO* (SXiO), introduced by Bitansky et al. [18]. SXiO requires that the time to obfuscate a circuit is slightly smaller than the truth table of the circuit. More formally, for a function $c$, we say that $c$-SXiO is a $(t, \ell)$-compressing obfuscator with $t(s, n) = \ell(s, n) = c(n) \cdot \mathsf{poly}(s)$. Similar to the above case, when there exists some constant $\epsilon > 0$ such that $c(n) = 2^{n(1-\epsilon)}$, we denote this simply by SXiO. Bitansky et al. [18] showed that SXiO and any public-key encryption, both with sub-exponential security, imply iO. This was strengthened by Kitigawa et al. [69], who showed that SXiO and any one-way function, with sub-exponential security, imply iO.

These two settings of parameters have seemingly minor differences, but nevertheless, are not known to be equivalent. Moreover, as mentioned above, their known implications illustrate the richness of the world of compressing obfuscation, and indicate that efficiency is a fundamental property of obfuscation. Since the regime of parameters for compressing obfuscation is somewhat non-standard (especially, the distinction between time and output length in XiO), it has not received adequate attention, and as a result we know very little about it.

In this work, we provide a systematic study of compressing obfuscation as an independent cryptographic primitive, and thus characterize the extent to which efficiency plays a role in obfuscation.

## 1.1 Our Results

Our results span a wide range of topics concerning compressing obfuscation, including limitations of its power, existence in an information-theoretic setting, constructions for limited classes of functions, and correctness amplification.

**XiO vs. PKE.** We start by exploring the power of XiO as an independent cryptographic primitive. One the one hand, we know that when combined with LWE it implies full-fledged iO (which in turn implies almost all cryptographic primitives). On the other hand, as opposed to iO [70], we do not even know whether XiO by itself[2] implies one-way functions — the most basic cryptographic primitive.

One of the original applications of obfuscation, which was proposed by Diffie and Hellman back in 1976 [43], is to transform private-key encryption into public-key encryption. When combined with one-way functions, iO can be used to perform such a transformation, as shown by [45, 90]. This transformation can also be obtained starting from sub-exponentially secure SXiO and one-way functions [69].

This raises the same question regarding XiO: Can it bridge the gap between the world of private-key cryptography and that of public-key cryptography? We provide evidence that it cannot, and thus show a concrete lower bound on its potential power.

**Theorem 1.1** (informal)**.** *There is no fully black-box construction of a perfectly correct key-agreement protocol from one-way functions and perfectly correct* $2^{(1-\epsilon)n}$*-XiO for* any *constant* $\epsilon > 0$*, even with sub-exponential security.*

Our result follows the extended black-box model of [48, 49], and in particular holds even if the XiO scheme can obfuscate oracle-aided circuits which contain *both* XiO and one-way function gates. This model is stronger than the one considered in [10, 11, 16], in which the obfuscator is allowed to obfuscate circuits with only one-way function gates. By allowing circuits to contain all possible oracle gates, our framework captures the flavor of non-black-box constructions which give public-key encryption from one-way functions and either iO [90] or SXiO [18, 32, 71, 69]. Thus, our result is one of the strongest forms of the classical separation between one-way functions and

[2] Assuming any average- or worst-case hardness assumption. This is necessary as XiO exists unconditionally if $\mathsf{P} = \mathsf{NP}$.

pubic-key encryption due to Impagliazzo and Rudich [65]. We note that our result does not separate *imperfectly* correct key-agreement from (perfectly correct) XiO and one-way functions.

Previously, by combining [10, 18], a related result follows but in a significantly weaker black-box model and for XiO with somewhat weak compression. Concretely, [10] showed a separation of perfect key-agreement from imperfect private-key FE in a black-box model where one can obfuscate functions that have *only* one-way function gates, and [18] showed a black-box construction of $2^{n/2}$-XiO from such private-key FE. This implies a separation from $2^{(1-\epsilon)n}$-XiO where $0 < \epsilon \leq 1/2$, and when only allowing XiO to obfuscate circuits containing one-way function gates.

**Statistical security.** Our result that it is unlikely that key-agreement can be constructed from XiO and one-way functions can be viewed as "good news", as it hints that XiO is a somewhat "weak" primitive, and therefore it might be possible to base its existence on well-studied assumptions. In fact, it might even be possible that compressing obfuscation exists unconditionally (even if $\mathsf{P} \neq \mathsf{NP}$). Toward this end, we show almost matching upper and lower bounds for the existence of compressing obfuscation with statistical security, both for the case of perfect correctness and that of approximate correctness. Our results show tight connections between compressing obfuscation and various concepts in complexity theory and learning and thus we view this as one of the central takeaways of this work.

For the case of approximate correctness, we show a $2^{n^\epsilon}$-SXiO for $\epsilon > 0$ for small classes of circuits (such as $\mathsf{AC}^0$). On the other hand, we show that such an obfuscator cannot exist for larger classes of circuits that contain a (puncturable) PRF, unless $\overline{\mathsf{SAT}} \in \mathsf{AM}[2^{n^\epsilon}]$, where $\overline{\mathsf{SAT}}$ is the problem of deciding whether a formula is unsatisfiable and $\mathsf{AM}[t(n)]$ is the class of all languages on instances of size $n$ that have an $\mathsf{AM}$ protocol in which the running time of the verifier and the message sizes are at most $t(n)$.

**Theorem 1.2** (informal)**.** *There exists a statistically secure and approximately correct $2^{n^\epsilon}$-SXiO for $\mathsf{AC}^0$ and $\epsilon > 0$. On the contrary, unless $\overline{\mathsf{SAT}} \in \mathsf{AM}[2^{n^\epsilon}]$, there is no such obfuscator for any class that contains a (puncturable) PRF.*

This result naturally leads to the question of whether we can get a similar statement for the case of perfect correctness. We are unable to get such a result for SXiO, but we do get it for XiO, albeit with worse compression.[3]

**Theorem 1.3** (informal)**.** *There exists a $2^{n(1-\epsilon)}$-XiO for $\epsilon \in 1/\mathsf{poly}\log(n)$ with statistical security and perfect correctness for $\mathsf{AC}^0$.*

Ruling out statistically secure XiO with any compression is left as an open problem. We do show that unless $\overline{\mathsf{SAT}} \in \mathsf{AM}[2^{c(1-\epsilon)n}]$ for a universal constant $c \in \mathbb{N}$, there is no statistically secure and perfectly correct $2^{n(1-\epsilon)}$-SXiO for $\mathsf{AC}^0$ (see Theorem 6.2). It is known, by the recent result of Williams [92], that $\overline{\mathsf{SAT}} \in \mathsf{AM}[\tilde{O}(2^{n/2})]$. However, it might be that for larger values of $\epsilon$ (such as $\epsilon = 1 - (0.1/c)$ or even $\epsilon = 1 - o(1)$) it holds that $\overline{\mathsf{SAT}} \notin \mathsf{AM}[2^{c(1-\epsilon)n}]$.

The positive results are based on classical (PAC) learning algorithms [91, 78] and the circuit compression algorithm of [37]. Both negative results above rely on and extend the following analogous arguments from the iO literature [56, 29]. The first is of Goldwasser and Rothblum [56] who showed that statistical iO with perfect correctness cannot exist unless $\mathsf{NP} \subseteq \mathsf{SZK}$. The second is of Brakerski, Brzuska, and Fleischhacker [29] who extended the result of [56] to handle statistical iO with *approximate* correctness by showing that unless $\mathsf{coNP} \subseteq \mathsf{AM}$, it cannot exist (assuming additionally one-way functions).

[3]The obfuscator we get is weak due to two reasons. First, the class for which we obtain XiO does not contain (puncturable) PRFs and thus is not sufficient for known transformations to iO. Second, the compression we achieve is not enough for cryptographic applications.

**Correctness amplification.** Our results above suggest that approximate correctness might be easier to achieve than perfect correctness, in an information theoretic setting. Is this the case also in the computational setting? To address this question, we show a transformation from approximately correct XiO to perfectly correct XiO, assuming the original XiO applies to a large enough class of circuits. This transformation achieves optimal parameters and only incurs polynomial security loss, indicating that correctness is not the bottleneck in constructing XiO from standard assumptions.

**Theorem 1.4** (informal)**.** *If there exists an XiO scheme for all polynomial size circuits which is correct with probability* $(1/2 + 1/\mathsf{poly})$ *over the the inputs and the obfuscation, then there exists a perfectly correct XiO scheme, assuming polynomially-secure LWE and NIZKs.*[4]

Prior to this result, there were no correctness amplification procedures for XiO which required only polynomial security or achieved optimal parameters. Correctness amplifications for related primitives, such as those of [22, 3] for iO, do not apply to XiO, since they involve a random self-reducibility step which inherently requires running the obfuscator on polynomial-size inputs. The transformation of Bitansky et al. [17] shows how to transform an XiO which is correct with probability 0.99 over the inputs and the obfuscation to a weak notion of functional encryption. This notion of functional encryption was known to imply a relaxed notion of XiO, namely, XiO with preprocessing [75]. Our transformation works for a much weaker notion of correctness (as opposed to .99) and results in full-fledged, perfectly correct XiO (as opposed to XiO with preprocessing).

Technically, our regime of parameters introduces many difficulties which require us to tailor a construction that is based on a delicate combination of various types of error-correcting codes together with cryptographic primitives (inspired by [83]).

While we show this transformation for the case of XiO, our result extends naturally to the case of SXiO. In particular, we can obtain perfectly correct XiO from the transformation, or SXiO which is correct on all but a negligible fraction of obfuscations.

**Universal construction.** Using our correctness amplification procedure, we obtain a universal construction of an XiO (resp. SXiO), assuming only the mere existence of XiO (resp. SXiO) with *polynomial* security and only (very weak) approximate correctness. For XiO, the resulting universal construction satisfies perfect correctness. Note that in the context of iO, perfect correctness is known to be achievable using only derandomization assumptions [23]. Our result is obtained by adapting the robust combiner of Ananth et al. [2] to the setting of XiO (resp. SXiO) and then using our correctness amplification transformation.

## 1.2 Related Work

**Universal construction and robust combiners.** It was shown in [60] that, in general, a robust combiner implies the existence of a universal construction. A robust combiner for a cryptographic primitive takes several candidate constructions of the primitive and outputs one construction that is as good as any of the input constructions (see also [63, 64]). A combiner for encryption appears already in [12], and perhaps the most known universal construction is that of one-way functions, due to [72].

Combiners for obfuscation were given in [44, 2, 3]. The work of [2] shows a robust combiner for indistinguishability obfuscation with sub-exponential security loss, and assuming either LWE or DDH. The work of [3] removes the sub-exponential assumption, but does not go all the way to iO—it shows a transforming combiner from candidates for indistinguishability obfuscation of which one of them is polynomially secure to a secure functional encryption scheme.

[4] Using the recent work of [68], we believe that the assumption on NIZKs can be removed. We leave this modification to future work.

**Existence of iO.** Mahmoody et al. [81] showed that iO cannot be based on random oracles or on constant degree multilinear maps (in a black-box way). Garg et al. [48] showed that iO cannot be constructed from any type of encryption that has an "all-or-nothing" type of security (as in PKE or Witness Encryption). Lastly, Garg et al. [49] studied the minimal compactness needed from a functional encryption scheme to imply iO, and gave matching constructions, following [4, 21]. In both [48, 49], the results hold even if the primitives upon which iO cannot be based can receive circuits containing gates for each of the primitive's subroutines.

Limitations on the power of iO were studied by Asharov and Segev [10, 11] and by Bitansky, Degwekar and Vaikuntanathan [16]. So far, we know that iO and one-way functions do not imply collision-resistant hash functions [10], domain-invariant one-way permutations [11], and hardness in $\mathsf{NP} \cap \mathsf{coNP}$ [16]. Also, iO and one-way permutations do not imply hardness in $\mathsf{SZK}$ [16].

**Relaxations of iO.** In addition to (S)XiO, another relaxation of iO is *decomposable obfuscation* (dO), which was recently introduced by Liu and Zhandry [79]. Decomposable obfuscation relaxes the security requirement of iO by requiring that obfuscations of circuits which satisfy a new notion of functional equivalence are indistinguishable. In particular, it is efficient to verify if two circuits satisfy their notion of functional equivalence, unlike traditional functional equivalence. This is similar to the case of XiO, because it is applied on circuits with only logarithmic input size for polynomial time applications. In [79], they question whether iO with efficiently verifiable functional equivalence implies public-key encryption. In fact, they have to assume the existence of public-key encryption for all the applications of dO that they show which imply public-key encryption. As mentioned above, we show a separation from XiO and OWFs to public key encryption. Therefore, our result serves as further evidence to the hypothesis that (non) efficiently checkable functional equivalence is one of the key factors which distinguishes iO from notions like XiO and dO.

**Compressing primitives.** Recently, compressing witness encryption (WE) was studied by Brakerski et al. [30]. Witness encryption, introduced by Garg et al. [46], allows encrypting a message relative to a statement $x \in L$ for a language $L \in \mathsf{NP}$ such that anyone holding a witness to the statement can decrypt the message, but if $x \notin L$, then it is computationally hard to decrypt. A compressing WE is such that the encryption time (and thus size) is less than the time it takes to solve the $\mathsf{NP}$ instance. Brakerski et al. showed that such a WE scheme can be constructed under "standard" assumptions (such as LWE or bilinear maps with sub-exponential security). This is in sharp contrast to SXiO (or even XiO).

### 1.3 Paper Organization

We proceed with a technical overview of our results. Then, in Section 3 we overview our definitions. In Section 4 we show our correctness amplification, and in Section 5 we prove our impossibility result of construction key-agreement protocol from XiO and OWFs. In Section 6 we present our positive and negative regarding statistically secure XiO.

## 2 Technical Overview

In this section we provide a high level overview of our results. We start with the correctness amplification (and its application to universal constructions) in Section 2.1. We proceed with the limitations on the power of XiO in Section 2.2, and conclude with our constructions and impossibilities of statistically secure XiO in Section 2.3.

## 2.1 Correctness Amplification

Our correctness amplification for XiO is a transformation from an approximately correct XiO scheme to an XiO scheme that is perfectly correct. Here, by approximately correct, we mean an XiO scheme which is correct with probability $(1/2+1/\mathsf{poly})$ over the inputs and the obfuscation, and by perfectly correct, we mean an XiO scheme which is correct on all inputs and all obfuscations with probability 1. The starting point for our correctness amplification is the transformation of Bitansky et al. [17], which transforms an XiO scheme which is correct with probability .99 over the obfuscation and the inputs to a functional encryption (FE) scheme which is correct on all inputs (with all but negligible probability). At a high level, FE is a type of encryption which enables generating functional keys, such that decryption of a ciphertext corresponding to a message $m$ with a functional key for a circuit $C$ results in $C(m)$. The hope is that if we can adapt the [17] transformation to our case, then we can attempt to transform the correct FE back to XiO.

**From approximately correct XiO to correct FE.** In [17], they first observe that by averaging and standard BPP-type amplification, their XiO scheme can be amplified to one which is correct with probability .9 *only over the inputs*. Then, they transform this XiO to a correct FE using an error-correcting code, as follows. To encrypt a message $m$, they obfuscate a circuit $G_m$ which, on input $i$, outputs an encryption of $(m, i)$ using a succinct functional encryption scheme $\mathsf{sFE}$, that exists based on LWE [55]. Call the resulting obfuscated circuit $\widetilde{G}_m$. To generate a functional key for a circuit $C$, they generate an $\mathsf{sFE}$ functional key for a circuit $C'$ that on input $(m, i)$ outputs the $i$th bit of $\mathsf{ECC}(C(m))$, where $\mathsf{ECC}$ is an error-correcting code. To decrypt, they first evaluate the obfuscated circuit $\widetilde{G}_m$ on every input $i$ to obtain a list of encryptions of $(m, i)$ for all $i$. Then, they use the $\mathsf{sFE}$ functional key to decrypt each of these encryptions and finally, decode the result.

The reason why this is enough for [17] is that, first, by the BPP amplification, they obtain correct encryptions of $(m, i)$ for a .9 fraction of $i$'s, with all but negligible probability over the obfuscation. This lets them calculate $(\mathsf{ECC}(C(m))_i$ for a *large* $(\gg 3/4)$ fraction of the $i$'s. Second, they rely on the error-correcting code which, given $(\mathsf{ECC}(C(m))_i$ for *many* $(\gg 3/4)$ $i$'s, can recover $C(m)$.

In our case, a natural attempt would be to replicate their first step and then use an error-correcting code with better parameters for the second step. However, this approach fails: we are only guaranteed correctness with probability $(1/2 + 1/\mathsf{poly}(\lambda))$ over the obfuscation and the inputs, which is not enough for averaging and BPP-type amplification. Nevertheless, the framework of [17] is still a convenient starting point for us.

Our first challenge is to obtain every bit of the encryption of $(m, i)$ for sufficiently many $i$'s. One idea is to apply an error-correcting code to the output of $G_m$, so that for any index $i$ for which $G_m$ correctly outputs enough of the bits of the encryption of $(m, i)$, we can decode successfully. While this is not possible for our regime of parameters using classical binary error-correcting codes, this is achievable with binary *list-decodable* codes, which output a list of possibilities upon decoding a codeword, rather than a unique decoding. Therefore, we modify the circuit $G_m$ to output a list-decodable encoding of the encryption of $(m, i)$, one bit at a time, which will be decoded at decryption time. This introduces the complication that list-decoding gives many possibilities for the encryption of $(m, i)$ for each $i$. To address this, we employ a combination of NIZK proofs and commitments which enable us to uniquely decode from the decoded list. At a high level, we impose the requirement that in addition to the ciphertext of $(m, i)$, the circuit $G_m$ on input $i$ must output a NIZK proof certifying that the ciphertext is correct. This ensures that we obtain $\mathsf{sFE}$ encryptions of $(m, i)$ for a noticeable fraction of the inputs $i$. Thus, we have replaced the BPP-type amplification of [17] with list-decodable codes, NIZK proofs, and commitment schemes.

After this change, we have that for a noticeable (but small, say 1%) fraction of the $i$'s, we obtain a correct encryption of $(m, i)$. If we decrypt this with the $\mathsf{sFE}$ secret key of [17], we would hope to obtain $(\mathsf{ECC}(C(m)))_i$ for enough $i$'s such that $\mathsf{ECC}$ can successfully decode to $C(m)$, but this

does not quite work because we only have a very small fraction of correct encryptions. Indeed, no (binary) error-correcting code can recover from more than 50% error! To overcome this, we notice that we have additional information (thanks to the NIZK) – we know exactly for which $i$'s we obtained correct $\mathsf{sFE}$ encryptions of $(m, i)$. Therefore, we replace the error-correcting code in the [17] construction with a code that can recover from a high fraction (say 99%) of *erasures*. To obtain optimal parameters, this requires us to have $\mathsf{sFE}$ output alphabet symbols rather than bits, but this does not impact the correctness of the scheme. Combining these two steps, we obtain an FE scheme with amplified correctness. As far as we know, this combination of list-decodable codes and erasure-correcting codes is novel to this work.

These techniques nearly work, with the caveat that our first step only gives us the correct encryptions of enough $(m, i)$ when the obfuscator uses "good" random coins. Nevertheless, this can be remedied by using BPP-type amplification and leveraging the fact that our FE scheme always decrypts to $\perp$ or to the correct output, $C(m)$. Therefore, this results in an FE scheme which is correct for all inputs with all but negligible probability.

**From correct FE to correct XiO.** The only remaining step is to transform the FE back to XiO. The FE scheme we obtain from the above transformations is *weakly sublinear compact*, a weak notion of compactness which does not suffice for known transformations to XiO without assuming sub-exponential security. FE with weak sublinear compactness has the property that while the encryption time is proportional to the circuit size of circuits supported by the scheme, the ciphertext lengths are compact. We take advantage of this by having an obfuscation consist of many "short" encryptions, which exactly captures the requirement that the obfuscator has a long running time but a nontrivial output length.

In more detail, to obfuscate a circuit $C$, we encrypt a circuit $C_x$ for each $x \in \{0,1\}^{n/2}$, where $C_x(\cdot) = C(x\|\cdot)$. Then, we generate a functional key $\mathsf{sk}$ for a circuit $T$, which, given a circuit on $n/2$ bits, outputs its truth table. The ciphertexts and functional key serve as our obfuscation, which gives the desired efficiency for XiO exactly because of the weak compactness of FE. To evaluate the obfuscation on an input $x = x_1\|x_2$, we use FE to decrypt the ciphertext corresponding to $C_{x_1}$ with $\mathsf{sk}$, and select the element of the truth table corresponding to $x_2$. This transformation yields a correct and secure XiO scheme, in which for any circuit $C$ and every input $x$, it holds that the obfuscation of $C$ at the point $x$ agrees with $C(x)$ with all but negligible probability.

In the technical section, we present the full construction in a more streamlined manner. In particular, we compose the XiO to FE transformation with the FE to XiO transformation described above, which yields a transformation from approximately correct XiO to XiO that is correct on any input with all but negligible probability over the randomness of the obfuscator.

Given an XiO which is correct on any input with all but negligible probability, we can then apply another BPP-style transformation (this time we apply parallel repetitions and then take the majority vote) to get an obfuscator that for all but negligible fraction of the obfuscations the obfuscated circuit completely agrees with the input circuit. To conclude our correctness amplification, we observe that the running time for XiO allows the obfuscator to compute the truth table of the circuit it obfuscates. Therefore, we modify the obfuscator to check if an obfuscation $\widetilde{C}$ of a circuit $C$ is correct by running over all inputs. If $\widetilde{C}$ agrees with $C$, then $\widetilde{C}$ is used as the obfuscation, and if not, we simply output $C$ in the clear. This takes advantage of the running time of XiO and incurs only a negligible loss in security, resulting in a perfectly correct XiO.

### 2.1.1 A Universal Construction

An important application of correctness amplification is a universal construction. We show a universal construction for XiO (resp. SXiO) by combining our correctness amplification with the results of [2].

A universal construction for a primitive can be obtained via a *robust combiner* for that primitive, which is a transformation that takes several candidate constructions of the primitive and outputs one construction that is as good as any of the input constructions. It is robust in the sense that it should work even if some of the candidates have weak correctness guarantees, have bad running times, etc. A universal construction is then acquired by enumerating over all possible candidates while making sure not to be "fooled" by bad faulty candidates so that we end up with a correct candidate. Thus, it is guaranteed that the resulting candidate is correct and secure.

We observe that a *combiner* (i.e, a secure candidate assuming one exists) for XiO (resp. SXiO) can be obtained by adapting the construction for iO of Ananth et al. [2] which further relied on LWE. In the case of iO, their construction, on input circuit $C$, obfuscates a variant of $C$ that has the same input domain as $C$. In the security proof, they go "input-by-input" over this obfuscated circuit which results in a sub-exponential security loss. We notice that, in the case of XiO (resp. SXiO), the number of inputs in the above obfuscated circuit is at most logarithmic, so the very same proof can be carried out, losing only a polynomial term. Then, to make the combiner robust we use our correctness amplification procedure. This results in a universal construction of perfect XiO (resp. imperfect SXiO), assuming the existence of XiO (resp. SXiO) with very weak correctness.

## 2.2 Impossibility of Key-Agreement

To illustrate the difference between the power of compressing obfuscation and iO, we revisit one of the primary applications of iO—transforming a private-key scheme into a public-key one. In the context of iO, this transformation is performed by obfuscating the encryption circuit of a private-key encryption scheme, while embedding the symmetric secret key into the circuit. The public key is then simply the obfuscated circuit. In order to encrypt a message $m$, one has to choose randomness $r$ and run the obfuscated circuit on $(m, r)$ to obtain the ciphertext $c$. An important property of this construction is the ability to obfuscate circuits with "hardwired cryptography", e.g., the evaluation circuit of a pseudorandom function with a hardwired PRF key.

Since XiO is efficient only when obfuscating circuits with logarithmic size input, one cannot use the above approach with XiO even when the message space is limited to a single bit. Given the public key, the adversary can learn the entire truth table of the obfuscated circuit by enumerating over all inputs, thereby breaking the secrecy of the underlying message. Our proof formalizes this intuition, and shows that other attempts at such a transformation cannot succeed. We formalize this using a black-box separation, showing that no perfectly complete bit-agreement protocol can be constructed from perfectly correct XiO and one-way functions.

**Modeling non-black-box constructions.** Constructions that are based on indistinguishability obfuscation are almost always *non-black-box* in the underlying primitives. In the example above, the circuit being obfuscated is the encryption algorithm of a private-key encryption scheme and thus contains a specific circuit representation of the underlying one-way function as a sub-circuit. More complex constructions also use techniques which require obfuscating a circuit which itself may obfuscate smaller circuits (and evaluate smaller obfuscations).

To capture these types of constructions, we extend the framework of Asharov and Segev [10, 11], which enables the obfuscator to run on *oracle-aided* circuits, i.e., circuits that might contain oracle gates. In this manner, the specific representation of the one-way function in the example above is replaced by an oracle gate, which allows the construction to be black-box relative to the one-way function. While the results in [10, 11] hold relative to an obfuscator for circuits which can only contain one-way function gates, our separation allows circuits to contain both XiO and one-way function gates. This captures the known techniques to obtain public-key encryption from either iO or SXiO, and is similar to the class of constructions captured in the framework of [48, 49]. We refer to [10, 11] for details regarding this model (see also [16]), and for examples of how it captures

common techniques such as the punctured programming technique of Sahai and Waters [90] and its variants.

**The oracle.** Our result is obtained by presenting an oracle $\Gamma$ relative to which the following properties hold: (1) there exists a one-way function $f$; (2) there exists a perfectly-correct, exponentially-secure XiO scheme $\mathsf{xi}\mathcal{O}$ for all oracle-aided circuits $C^{\mathsf{xi}\mathcal{O},f}$; (3) for any perfectly complete bit-agreement protocol between two parties, there exists an eavesdropping adversary that makes polynomially many queries to the oracle $\Gamma$ and succeeds to recover the bit from the transcript of the interaction. Our oracle consists of three functions, similar to that of [11]: (1) a random function $f$ that will serve as the one-way function; (2) a random length-increasing function $\mathcal{O}$ that will serve as the obfuscator (an obfuscation of an oracle-aided circuit $C$ is a "handle" $\widehat{C} = \mathcal{O}(C, r)$ for a random string $r$), and (3) a function $\mathcal{E}$ that enables evaluations of obfuscated circuits: given some obfuscated circuit $\widehat{C}$ and an input $x$, the function $\mathcal{E}$ looks for the lexicographically first pair $(C, r)$ for which $\mathcal{O}(C, r) = \widehat{C}$ and returns $C^{\Gamma}(x)$. Note that if $C$ is some circuit of size $s$, it can only make oracle calls to $\Gamma$ on inputs smaller than $s$, and thus the above definition is not circular.

The main difference in modeling XiO as an oracle rather than iO as in [11] is the expansion factor of the oracle $\mathcal{O}$. In order to capture compressing obfuscation, the expansion factor that we use is (sub-)exponential in the input size of the circuit $C$. While this modification is somewhat minor in syntax, it has a major effect – if the expansion factor is "small" then it is possible to construct a *polynomial time* key-agreement protocol relative to such an oracle (following the construction of Sahai and Waters [90]), whereas for a larger expansion factor this becomes impossible. As for the existence of one-way functions and indistinguishability of obfuscated circuits, we derive these almost for free from [11].

In what follows, we first discuss how to break a perfectly complete key-agreement protocol relative to a random oracle as a warmup. We then discuss the challenges when dealing with our (more structured) oracle, and discuss why our approach does not work for iO.

**Separating key-agreement from a random oracle.** As a warmup, we first give an overview of the result of Impagliazzo and Rudich [65] and Brakerski et al. [31], who show that for any two polynomial time oracle-aided algorithms $\mathcal{A}$ and $\mathcal{B}$, if $\langle \mathcal{A}^f, \mathcal{B}^f \rangle$ implements a perfectly-correct bit-agreement protocol for all functions $f$, then there exists an oracle-aided algorithm $E$ such that for any function $f$ learns the agreed bit with probability 1 by making only a polynomial number of oracle queries to $f$. The adversary $E$ is given a transcript $T$ which is a result of an interaction of $\mathcal{A}$ and $\mathcal{B}$ relative to some oracle $f$, and is required to find the key $k^\star$ that $\mathcal{A}$ and $\mathcal{B}$ agreed on. Denote by $r^\star_{\mathcal{A}}$ (resp. $r^\star_{\mathcal{B}}$) the randomness used by $\mathcal{A}$ (resp. $\mathcal{B}$) in the real interaction that produced $T$. The adversary $E$ initializes a set of queries/answers $Q$, which will contain the actual queries made by $E$ to the true oracle $f$. It also initializes a multiset $K = \emptyset$, and repeats the following polynomially many times:

- **Simulation:** $E$ simulates an oracle $f'$ that is consistent with $Q$ (i.e., $f'(w) = f(w)$ for every $w \in Q$), and randomness $r'_{\mathcal{A}}, r'_{\mathcal{B}}$ such that the interaction $\langle \mathcal{A}^{f'}(r'_{\mathcal{A}}), \mathcal{B}^{f'}(r'_{\mathcal{B}}) \rangle$ (i.e., running the protocol with respect to the function $f'$ with randomness $r'_{\mathcal{A}}$ for $\mathcal{A}$ and $r'_{\mathcal{B}}$ for $\mathcal{B}$) results in the transcript $T$ and key $k'$. $E$ adds $k'$ to $K$.
- **Update:** $E$ asks $f$ for all queries made to $f'$ by $\mathcal{A}$ or $\mathcal{B}$ in the simulation that are not already in $Q$, and updates the set $Q$.

At the end of the attack, $E$ outputs the majority value in $K$. The proof then relies on the following observation: In each iteration, either (1) in the update phase, $E$ finds at least one new query that is also made by either $\mathcal{A}$ or $\mathcal{B}$ during the real interaction with the function $f$ that produced the transcript $T$; or (2) $E$ adds the real key $k^\star$ to $K$.

Intuitively, if (1) does not hold, then the perfect correctness of the bit-agreement protocol guarantees that (2) holds. In particular, in that case it is possible to construct a "hybrid" oracle $\widetilde{f}$ that behaves like $f$ in the real execution of $\mathcal{A}$, i.e., $\mathcal{A}^f(r^\star_{\mathcal{A}})$, and behaves like $f'$ in the simulated evaluation of $\mathcal{B}$, i.e., $\mathcal{B}^{f'}(r'_{\mathcal{B}})$. According to this hybrid oracle, an execution of $\mathcal{A}$ with randomness $r^\star_{\mathcal{A}}$ and an execution of $\mathcal{B}$ with randomness $r'_{\mathcal{B}}$ would result in the transcript $T$, $\mathcal{A}$ would output $k^\star$ (as in the real execution) and $\mathcal{B}$ would output $k'$ (as in the simulation). Perfect correctness then tells us that $k^\star = k'$. This hybrid oracle can be constructed since the intersection of the set of queries made in the simulated execution and those made in the real execution is already contained in $Q$, and therefore there are no contradicting queries (i.e., queries $w$ that appear in both executions for which $f(w) \neq f'(w)$). As the number of oracle queries $\mathcal{A}$ and $\mathcal{B}$ make during the execution of the protocol is some polynomial $q$, the majority value in $K$ is guaranteed to be the correct key after $2q + 1$ iterations.

**Attacking key-agreement relative to our oracle.** We extend the attack described above relative to our oracle $\Gamma$, which is a significantly more structured than a random oracle and therefore raises several challenges. Recall that our oracle $\Gamma$ consists of a three functions $f$, $\mathcal{O}$, and $\mathcal{E}$, that are dependent. Following the above template, we construct an adversary that simulates an execution that produces the transcript $T$ with some simulated oracle $\Gamma' = (f', \mathcal{O}', \mathcal{E}')$. There are three main challenges with this approach:

1. The first challenge is to show that $\mathcal{A}$ and $\mathcal{B}$ cannot gain "extra" information from oracle queries that are not in the intersection of their query sets. In particular, in the case of a random oracle, the shared information between $\mathcal{A}$ and $\mathcal{B}$ can be recovered completely from their shared oracle queries and the transcript $T$. In our setting, since the oracles $f$, $\mathcal{O}$, and $\mathcal{E}$ have dependence, this may not be the case.

2. The second challenge is due to the fact that queries made by $\mathcal{A}$ and $\mathcal{B}$ could cause "hidden" oracle queries. Since we allow obfuscated circuits to contain oracle gates, this could occur when obfuscated circuits are evaluated by $\mathcal{A}$ and $\mathcal{B}$. In particular, the output of the evaluation could reveal query-answer pairs on queries that were never directly asked by $\mathcal{A}$ or $\mathcal{B}$. Thus, we must show that $\mathcal{A}$ and $\mathcal{B}$ cannot indirectly learn too many query-answer pairs this way.

3. The third challenge is to show that a hybrid oracle $\widetilde{\Gamma} = (\widetilde{f}, \widetilde{\mathcal{O}}, \widetilde{\mathcal{E}})$ can be constructed from the two sets of queries, i.e., from the simulated execution and the real execution. As an example, suppose there is a query $\mathcal{E}(\widehat{C}, x)$ that is performed in the real execution and a different query $\mathcal{E}'(\widehat{C}, y)$ that appears in the simulated execution. Such two queries raise a challenge for constructing a hybrid oracle $\widetilde{\mathcal{E}}$ which is consistent with these two queries simultaneously. In order to see this, suppose that in the real execution, the lexicographically first pair $(C, r)$ for which $\mathcal{O}(C, r) = \widehat{C}$ is some pair $(C_1, r_1)$, and in the simulated execution the lexicographically first pair $(C, r)$ for which $\mathcal{O}'(C, r) = \widehat{C}$ is some pair $(C_2, r_2) \neq (C_1, r_1)$. As a result, $\mathcal{E}(\widehat{C}, x)$ in the real execution is mapped to $C_1^\Gamma(x)$, whereas $\mathcal{E}'(\widehat{C}, y)$ is mapped to $C_2^{\Gamma'}(y)$, but $C_1 \neq C_2$.

We solve the first challenge by adding additional oracle queries to the set of real queries that the parties make, which makes the dependence between the oracles more explicit. We solve the second challenge by showing that any oracle query can only cause polynomially many additional indirect queries. In particular, for a circuit $C^\Gamma$ of size $s$, any indirect queries are on circuits smaller than $s$. We use this in conjunction with the (sub-exponential) expansion factor of our oracle $\mathcal{O}$ to show that the number of indirect queries are bounded, and thus the adversary $E$ can learn any indirect queries that $\mathcal{A}$ and $\mathcal{B}$ learn by only making polynomially many additional queries.

As for the third challenge, interestingly, our proof does not completely solve it, and we do not fully control to which one of the two circuits $C_1$ or $C_2$ the hybrid oracle $\widetilde{\mathcal{E}}$ maps $\widehat{C}$. Nevertheless,

we design the adversary such that, whenever there is such a contradicting scenario between the real execution and the simulated execution, it must hold that $C_1$ and $C_2$ are functionally equivalent with respect to the hybrid oracle $\widetilde{\Gamma}$. Otherwise, i.e., when there is some input for which $C_1$ and $C_2$ do not agree, we claim that the adversary learns a new query that is associated with the real execution. As a consequence, $E$ learns the entire truth table of any obfuscated circuit $\widehat{C}$ that is associated with the real execution, which is possible due to the fact that querying the oracle $\Gamma$ on all inputs of $\widehat{C}$ results in polynomially many queries. Notably, for a different expansion factor of the oracle $\mathcal{O}$ (which results in iO and not XiO), this becomes an exponential number of queries, and the above attack fails.

### 2.3 Statistically Secure Compressing Obfuscation

This set of results is composed of two main parts. One is positive results showing that for small classes of circuits compressing obfuscation exists unconditionally. The other complements the constructions and shows that improvements in the above obfuscator, either in the compression factor or in the circuit class, will imply some nontrivial speedup for protocols solving $\mathsf{UNSAT}$. We have positive and negative results both for the case of perfect correctness and for the case of approximate correctness.

**Negative results.** First, we show that that approximately correct and statistically secure $2^{n^\epsilon}$-SXiO cannot exist unless $\mathsf{coNP} \subseteq \mathsf{AM}[2^{n^\epsilon}]$ for $\epsilon > 0$. Here, we follow on the approach of [29] from the world of iO. There, they show how to use iO and puncturable PRFs to create two circuits that differ at a single point but their obfucations (as random variables) are statistically far. Then, they use an algorithm that can distinguish these two distributions to solve Unique-SAT which then implies that $\mathsf{coNP} \subseteq \mathsf{AM}$ by a result of Mahmoody and Xiao [82]. We modify the argument to work with compressing obfuscation by making the two circuits receive only short inputs, and observe that the proof still goes through, but then solving Unique-SAT on short inputs (say of poly-logarithmic size). We then apply the result of Mahmoody and Xiao and finally obtain our result by scaling the parameters.

Second, we show that perfectly correct and statistically secure $2^{n(1-\epsilon)}$-SXiO cannot exist unless $\mathsf{coNP} \subseteq \mathsf{AM}[2^{(1-\epsilon)n}]$ (with large enough $0 < \epsilon < 1$). For this, we construct an $\mathsf{SZK}[2^{(1-\epsilon)n}]$ protocol for all $\mathsf{NP}$. In this protocol, the verifier, given $x \in L$ for a language $L$, chooses a bit $b$ uniformly at random and obfuscates a circuit that gets a witness $w$ as input, checks whether it is a valid witness for $x$ and if so, it outputs $b$ (otherwise it outputs $\perp$). This protocol can be shown to be honest-verifier statistical zero-knowledge with a verifier that runs in time $2^{(1-\epsilon)n}$ for $L$. This argument is reminiscent to the argument of [70, 56] in the context of iO. We then carefully apply the transformation of Okamoto [86] to translate this protocol into an (honest-verifier) SZK protocol for every language in $\mathsf{coNP}$. This implies that $\mathsf{coNP} \subseteq \mathsf{AM}[2^{(1-\epsilon)n}]$.

**Positive results.** We show that compressing obfuscators exists unconditionally for restricted classes of circuits such as $\mathsf{AC}^0$ (the class of all constant-depth circuits) and $\mathsf{Mon}$ (the class of all monotone functions). We again construct compressing obfuscators with perfect correctness and approximate correctness. The approximately correct obfuscators are obtained by running a classical (PAC) learning algorithm [91] on the given circuit and outputting the hypothesis. Using the most efficient learning algorithms for $\mathsf{AC}^0$ and $\mathsf{Mon}$, we obtain compressing obfuscators for these classes. This construction is aligned with the above impossibility that says that we are unlikely to be able to get such an obfuscator for classes that contain a (puncturable) PRF.

In the perfect correctness case, we use a different tool called a *circuit compression* algorithm [37]. In circuit compression one is given the truth table of a Boolean function $f$ computable by some *unknown* circuit from a known class of circuits, and the goal is to find in time $\mathsf{poly}(2^n)$ a circuit $C$

(not necessarily from the aforementioned family) computing $f$ so that the size of $C$ is less than the trivial circuit size $\approx 2^n$. We apply such an algorithm on circuits in $\mathsf{AC}^0$ and get an obfuscator with small compression.

# 3 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution $X$ we denote by $x \leftarrow X$ the process of sampling a value $x$ from the distribution $X$. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value $x$ from the uniform distribution over $\mathcal{X}$. For a randomized function $f$ and an input $x \in \mathcal{X}$, we denote by $y \leftarrow f(x)$ the process of sampling a value $y$ from the distribution $f(x)$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$.

Throughout the paper, unless otherwise specified, we denote the security parameter by $\lambda$. A function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer $N_c$ such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Two sequences of random variables $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are *computationally indistinguishable* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$.

When we deal with Boolean circuits, we parametrize them by their size $s$ and the number of inputs they accept $n$. As usual, the size of a circuit is defined to be the number of wires in it.

**Definition 3.1.** *For any functions $s(\cdot)$ and $n(\cdot)$, we define $\mathcal{C}^{s,n}$ to be the class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ for which for any $C \in C_\lambda$, the size of $C$ is at most $s(\lambda)$ and the input length of $C$ is at most $n(\lambda)$.*

**Definition 3.2.** *We define the following classes of circuits:*

- $\mathsf{P}^{\log}$*: the collection of circuit classes $\mathcal{C}^{s,n}$ for which $s$ is a polynomial, $n$ is a logarithmic function, and for which all circuits have one-bit outputs.*
- $\mathsf{P}$*: the collection of circuit classes $\mathcal{C}^{s,n}$ for which $s$ and $n$ are polynomials.*

**Definition 3.3.** *For an (uniform) algorithm $A$ with input $x$, we denote by $\mathsf{Time}[A(x)]$ an upper bound on the running time of $A$ on input $x$. We denote by $\mathsf{Outlen}[A(x)]$ an upper bound on the output length of $A$ when run on input $x$.*

## 3.1 Compressing Obfuscation

We define a general notion of compressing obfuscation, generalizing the definition of [75].

**Definition 3.4** (Functional equivalence)**.** *We say that two circuits $C$ and $C'$ are* functionally equivalent *and denote it by $C \equiv C'$ if they compute the same function (i.e., $\forall x : C(x) = C'(x)$).*

**Definition 3.5** (Compressing obfuscation)**.** *An $\alpha$-correct $(t, \ell)$-*compressing obfuscator *for the circuit class $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a pair of algorithms $(\mathsf{Obf}, \mathsf{Eval})$ with the following syntax:*

- $\widetilde{C} \leftarrow \mathbf{Obf}(1^\lambda, C)$**:** *The obfuscator receives the security parameter $1^\lambda$ and a circuit $C \in \mathcal{C}^{s,n}$ and outputs a circuit $\widetilde{C}$.*
- $\mathbf{Eval}(\widetilde{C}, x)$**:** *The evaluator receives a circuit $\widetilde{C}$ and an input $x$, and outputs a string $y$ or $\perp$.*
- $\boldsymbol{\alpha}$**-Correctness.** *For all $\lambda \in \mathbb{N}$, all $C \in \mathcal{C}_\lambda$, and all $x \in \{0,1\}^n$, it holds that*

$$\Pr_{\mathsf{Obf}}\left[\widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C) : C(\cdot) \equiv \mathsf{Eval}(\widetilde{C}, \cdot)\right] \geq \alpha(\lambda)$$

- **$(t, \ell)$-Compression.** *For all $\lambda \in \mathbb{N}$ and all $C \in \mathcal{C}_\lambda$, there exists a polynomial $\mathsf{poly}(\cdot)$ such that the running time and output size of $\mathsf{Obf}(1^\lambda, C)$ are bounded by $t(s,n) \cdot \mathsf{poly}(\lambda, n)$ and $\ell(s,n) \cdot \mathsf{poly}(\lambda, n)$, respectively. That is, for $n = n(\lambda)$ and $s = s(\lambda)$,*

$$\mathsf{Time}\left[\mathsf{Obf}(1^\lambda, C)\right] = t(s,n) \cdot \mathsf{poly}(\lambda, n), \qquad \mathsf{Outlen}\left[\mathsf{Obf}(1^\lambda, C)\right] = \ell(s,n) \cdot \mathsf{poly}(\lambda, n).$$

Several instantiations of $s(\cdot)$ and $\ell(\cdot)$ are of interest in this work. Fix $\epsilon > 0$. One setting is when the obfuscation size is exponential in $(1-\epsilon)n$, but the running time is exponential in $n$. The other setting is when both the running time and the output size are exponential in $(1-\epsilon)n$.

**Definition 3.6** (XO)**.** *An* exponentially-efficient obfuscator (XO) *for a class $\mathcal{C}^{s,n}$ of circuits is a $(t, \ell)$-compressing obfuscator with*

$$\ell(s,n) = 2^{n(1-\epsilon)} \cdot \mathsf{poly}(s), \qquad t(s,n) = \mathsf{poly}(2^n, s).$$

**Definition 3.7** (SXO)**.** *A* strong exponentially-efficient obfuscator (SXO) *for a class $\mathcal{C}^{s,n}$ of circuits is a $(t, \ell)$-compressing obfuscator with*

$$t(s,n) = \ell(s,n) = 2^{n(1-\epsilon)} \cdot \mathsf{poly}(s)$$

We provide two security definitions for a compressing obfuscator as above, along the lines of indistinguishability obfuscation [15, 45] and of virtual black-box obfuscation [15].

**Definition 3.8** (Indistinguishability obfuscation)**.** *An $\alpha$-correct* $(t, \ell)$-compressing obfuscator $O$ *is an indistinguishability obfuscator (iO) for the class $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if for any probabilistic polynomial-time distinguisher $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, \in \mathbb{N}$ and all $C_0, C_1 \in \mathcal{C}_\lambda$ with $C_0 \equiv C_1$, it holds that*

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

## 3.2 Correctness of Obfuscation

In addition to the notion of $\alpha$-correctness we gave above, we define three additional notions of correctness for obfuscation, as in [22].

**Definition 3.9** (Perfect correctness)**.** *An $\alpha$-correct obfuscator for a circuit class $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is* perfectly correct *if $\alpha(\lambda) = 1$ for all values of $\lambda$.*

**Definition 3.10** (Almost Perfect Correctness)**.** *An obfuscator $(\mathsf{Obf}, \mathsf{Eval})$ for a class of circuits $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is* almost perfectly correct *if for all $\lambda \in \mathbb{N}$ and all $C \in \mathcal{C}_\lambda$, it holds that*

$$\Pr_{\mathsf{Obf}}\left[\widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C) : \forall x \in \{0,1\}^n \colon C(x) = \mathsf{Eval}(\widetilde{C}, x)\right] \geq \alpha(\lambda).$$

**Definition 3.11** (Approximate Correctness)**.** *An obfuscator $(\mathsf{Obf}, \mathsf{Eval})$ for a class of circuits $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ is* $\alpha$-approximately correct *if for all $\lambda \in \mathbb{N}$ and all $C \in \mathcal{C}_\lambda$, it holds that*

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

### 3.3 Puncturable Pseudorandom Functions

**Definition 3.12** (Puncturable PRF [90])**.** *A puncturable pseudorandom function* $\mathsf{PRF}$ *is given by a tuple of efficient algorithms* $\mathsf{PRF} = (\mathsf{Key}, \mathsf{Punc}, \mathsf{Eval})$ *and a pair of computable functions* $\ell(\cdot)$ *and* $m(\cdot)$ *(where the* $\mathsf{PRF}$ *maps* $\ell$*-bit inputs to* $m$*-bit outputs) which satisfy the following conditions:*

- **Functionality preserved under puncturing:** *For every polynomial size set* $S \subseteq \{0,1\}^{\ell(\lambda)}$ *and for every* $x \in \{0,1\}^{\ell(\lambda)} \setminus S$*, we have that:*

$$\Pr\left[K \leftarrow \mathsf{Key}(1^\lambda), \widetilde{K}_{[S]} = \mathsf{Punc}(K, S) : \mathsf{Eval}(K, x) = \mathsf{Eval}(\widetilde{K}_{[S]}, x)\right] = 1.$$

- **Pseudorandomness at punctured points:** *For every probabilistic polynomial-time adversary* $\mathcal{A}$ *and every point* $x^* \in \{0,1\}^{\ell(\lambda)}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, it holds that*

$$\left|\Pr\left[\mathcal{A}(\widetilde{K}_{x^*}, \mathsf{Eval}(K, x^*)) = 1\right] - \Pr\left[\mathcal{A}(\widetilde{K}_{x^*}, U_{m(\lambda)}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

*where* $K \leftarrow \mathsf{Key}(1^\lambda)$*,* $\widetilde{K}_{x^*} = \mathsf{Punc}(K, x^*)$*, and* $U_{m(\lambda)}$ *denotes the uniform distribution over* $\{0,1\}^{m(\lambda)}$*.*

**Theorem 3.13** ([53, 26, 28, 67])**.** *Assuming the existence of one-way functions, for any computable functions* $\ell(\cdot)$ *and* $m(\cdot)$ *there exists a secure puncturable PRF family mapping* $\{0,1\}^{\ell(\lambda)}$ *to* $\{0,1\}^{m(\lambda)}$*.*

### 3.4 Non-Interactive Zero Knowledge

In this work, we consider NIZK proof systems in the CRS model which support proving and simulating multiple statements with the same CRS. We start with the definition of a non-interactive zero knowledge proof system as in [45].

**Definition 3.14.** *Let* $L$ *be a language with a relation* $R_L$*. Without loss of generality, assume that for any* $(x, w) \in R_L$*, it holds that* $|x| = |w| = n$*. A* non-interactive zero-knowledge proof system in the CRS model *consists of a tuple of PPT algorithms* $\mathsf{NIZK} = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *described as follows:*

- $\sigma \leftarrow \mathsf{Gen}(1^\lambda, 1^n)$*: The* $\mathsf{Gen}$ *algorithm takes as input the security parameter* $\lambda$ *and outputs the CRS* $\sigma$*.*
- $\pi \leftarrow \mathsf{P}(\sigma, x, w)$*: The prover algorithm* $\mathsf{P}$ *takes as input the CRS* $\sigma$*, a statement* $x$ *for the language* $L$*, and a witness* $w$*, and outputs a proof* $\pi$*.*
- $b \leftarrow \mathsf{V}(\sigma, x, \pi)$*: The verifier algorithm* $\mathsf{V}$ *takes as input the CRS* $\sigma$*, a statement* $x$*, and a proof* $\pi$*, and outputs a bit* $b \in \{0,1\}$*.*

*We require the following properties of the NIZK scheme.*

- **Perfect Completeness.** *There exists a negligible function* $\mathsf{negl}$ *such that for every* $x \in L$ *and* $w \in \mathcal{R}_L(x)$*, for all* $\lambda \in \mathbb{N}$*,*

$$\Pr\left[\sigma \leftarrow \mathsf{Gen}(1^\lambda, 1^n); \pi \leftarrow \mathsf{P}(\sigma, x, w) : \mathsf{V}(\sigma, x, \pi) = 1\right] = 1.$$

- **Statistical Soundness.** *For every (possible unbounded) adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}$ *such that for every* $\lambda \in \mathbb{N}$*:*

$$\Pr\left[\sigma \leftarrow \mathsf{Gen}(1^\lambda, 1^n) : (x, \pi) \leftarrow \mathcal{A}(\sigma) : \mathsf{V}(\sigma, x, \pi) = 1 \wedge x \notin L\right] \leq \mathsf{negl}(\lambda).$$

- **Computational Zero knowledge.** *There exists a pair of PPT simulators* $(S_1, S_2)$ *such that for every* $x \in L$*, every* $w \in R_L(x)$*, and every* $\lambda \in \mathbb{N}$*, the following two distributions are computationally indistinguishable:*

$$\{\sigma \leftarrow \mathsf{Gen}(1^\lambda, 1^n); \pi \leftarrow P(\sigma, x, w) : (\sigma, x, \pi)\}$$

$$\{(\sigma', \mathsf{aux}) \leftarrow S_1(1^\lambda, 1^n); \pi' \leftarrow S_2(\sigma', x, \mathsf{aux}) : (\sigma', x, \pi')\}$$

We require the NIZK proof system to support proving and simulating polynomially many theorems from one CRS. This is captured by the following definition, as in [52].

**Definition 3.15.** *Let* $L$ *be a language with relation* $R_L$*. A Multi-NIZK proof system* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *for* $L$ *in the CRS model is* multiple theorem computational zero knowledge *if for any polynomial* $m(\cdot)$*, there exists a pair of PPT simulators* $(S_1, S_2)$ *such that for any* $\lambda \in \mathbb{N}$ *and any* $\{x_i, w_i\}_{i \in [m(\lambda)]}$ *with* $w_i \in R_L(x_i)$ *and* $|x_i| = |w_i| = n$ *for all* $i$*, the following two distributions are computationally indistinguishable:*

$$\left\{\sigma \leftarrow \mathsf{Gen}(1^\lambda, 1^n); \pi_i \leftarrow P(\sigma, x_i, w_i)\ \forall i \in [m(\lambda)] : \left(\sigma, \{x_i, \pi_i\}_{i \in m[\lambda]}\right)\right\}$$

$$\left\{(\sigma', \mathsf{aux}) \leftarrow S_1(1^\lambda, \{x_i\}_{i \in m[\lambda]}); \pi'_i \leftarrow S_2(\sigma', x_i, \mathsf{aux})\ \forall i \in [m(\lambda)] : \left(\sigma', \{x_i, \pi'_i\}_{i \in m[\lambda]}\right)\right\}.$$

It is known that NIZKs satisfying multiple-theorem computational zero knowledge can be built from any NIZK proof system [52]. We stress that the CRS $\sigma$ is of length $\lambda$, and supports proving $\mathsf{poly}(\lambda)$ many statements of length $\mathsf{poly}(\lambda)$.

### 3.5 Commitment Schemes

For some of our constructions, we need a non-interactive commitment scheme such that commitments of different strings have disjoint support. Jumping ahead, we will use commitments in some encryption procedure that is not controlled by the adversary (i.e., it is honest). Therefore, we can relax the foregoing requirement and use non-interactive commitment schemes that work in the CRS (common random string) model (for ease of notation, we usually ignore the CRS).

**Definition 3.16.** *A* non-interactive commitment scheme *is a tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open})$ *described as follows:*

- $c \leftarrow \mathsf{Commit}(x, r)$*: The commitment algorithm* $\mathsf{Commit}$ *takes a message* $x \in \{0,1\}^n$ *for* $n = \mathsf{poly}(\lambda)$ *and a random string in* $\{0,1\}^\lambda$ *and outputs a commitment* $c$ *to* $x$*.*
- $x' \leftarrow \mathsf{Open}(c, r)$*: The opening algorithm takes as input a commitment* $c$ *and a random string* $r \in \{0,1\}^\lambda$*, and outputs a message* $x'$*.*

*We require the following properties of a commitment scheme.*

- **Hiding.** *For any* $x_1, x_2$*, the distributions* $\{\mathsf{Commit}(x_1, \mathcal{U}_\lambda)\}_{\lambda \in \mathbb{N}}$ *and* $\{\mathsf{Commit}(x_2, \mathcal{U}_\lambda)\}_{\lambda \in \mathbb{N}}$ *are computationally indistinguishable, where* $\mathcal{U}_\lambda$ *is the uniform distribution over* $\{0,1\}^\lambda$*.*
- **Binding.** *For any* $x_1, x_2 \in \{0,1\}^n$ *and* $r_1, r_2 \in \{0,1\}^\lambda$*,* $\mathsf{Commit}(x_1, r_1) \neq \mathsf{Commit}(x_2, r_2)$*.*

Commitment schemes that satisfy the above definition, in the CRS model, can be constructed based on any pseudorandom generator [85] (which can be based on any one-way functions [61]). We say that $\mathsf{Commit}(x, r)$ is the $\mathsf{commitment}$ to the value $x$ with the $\mathsf{opening}$ $r$.

### 3.6 Error-Correcting Codes

We review the definitions of error-correcting that are relevant to this paper. A code $C$ over an alphabet $\Sigma$ of size $q$ that has block length $n$, dimension $k$ and minimal distance $d$ is denoted as an $(n, k, d)_q$ code. A code $C$ can be thought of as a mapping from $\Sigma^k$ to $\Sigma^n$ such that every two outputs of the mapping differ in at least $d$ locations. The mapping procedure is sometimes referred to as the encoding function of $C$. The *relative distance* of $C$ is $d/n$ and the *rate* is $k/n$.

In this paper, we will use a Reed-Solomon code as an erasure-correcting code, which can recover from a small enough fraction of deleted symbols in the codeword.

**Theorem 3.17** (e.g., [58, Chapter 11])**.** *The Reed-Solomon error-correcting code is an $(n, k, n - k + 1)_q$ code for $k < n \leq q$ that can be uniquely decoded by a polynomial time algorithm from any fraction $e$ of erasures satisfying $en \leq n - k + 1$.*

We will also use binary error correcting codes. It is known that binary error correcting codes with unique decoding cannot correct a $1/2$ fraction of errors, so we will need the list-decoding relaxation that allows the decoder to output a (short) *list* of possible messages such that the correct message is one of them.

**Definition 3.18** (List decoding)**.** *A binary error-correcting code is $(e, L)$-list decodable if for any $c \in \{0,1\}^n$, there are at most $L$ codewords within distance $e \cdot n$ of $c$ and there is a polynomial time algorithm* decode *such that on input $c \in \{0,1\}^n$, outputs all such codewords.*

**Theorem 3.19** ([59, Corollary 4])**.** *For any integer $r$ and $\gamma > 0$, there exists a polynomial $p(\cdot)$ such that there exists a binary error correcting code of rate $r$ and block length $n$ where $n = O\left(r/\gamma^8\right)$, that is $(\frac{1}{2} - \frac{\gamma}{2}, p(n))$-list decodable.*

### 3.7 Functional Encryption

**Definition 3.20** (Functional Encryption [87, 25])**.** *A public-key functional encryption (FE) scheme for a class of circuits $\mathcal{C}^{s,n}$ is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *that behave as follows:*

- $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter $\lambda$ and outputs the master secret key* msk *and public key* pk*.*
- $\mathsf{sk}_C \leftarrow \mathsf{FE.Keygen}(\mathsf{msk}, C)$*: The key generation algorithm takes as input the master secret key* msk *and some circuit $C \in \mathcal{C}_\lambda$ and outputs the functional secret key $\mathsf{sk}_C$.*
- $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{pk}, m)$*: The encryption algorithm takes as input the public key* pk *and a message $m$ and outputs a ciphertext* cipher*.*
- $y \leftarrow \mathsf{FE.Dec}(\mathsf{sk}_C, \mathsf{ct})$*: The decryption algorithm takes as input the functional secret key $sk_C$ and ciphertext* ct *and outputs $y \in \{0,1\}^*$.*

*We require the following conditions to hold:*

- **Correctness:** *There exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$, every $C \in \mathcal{C}_\lambda$, and every message $m \in \{0,1\}^{n(\lambda)}$, we have that:*

$$\Pr\left[C(m) = \mathsf{FE.Dec}\left(\mathsf{FE.Keygen}(\mathsf{msk}, C), \mathsf{FE.Enc}(\mathsf{pk}, m)\right)\right] \geq 1 - \mathsf{negl}(\lambda),$$

  *where $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, and the probability is taken over the randomness of* FE.Setup*,* FE.Keygen *and* FE.Enc*.*

- **$q$-selective security:** *For every probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, every collection of $q$ circuits $C_1, \ldots, C_q \in \mathcal{C}_\lambda$, and ever pair of messages $m_0, m_1 \in \{0,1\}^{n(\lambda)}$ such that $C_i(m_0) = C_i(m_1)$ for all $i \in [q]$, it holds that*

$$\left| \Pr\left[\mathcal{A}\left(z, \mathsf{FE.Enc}(\mathsf{pk}, m_b)\right) = b\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

*where $z = (\mathsf{pk}, C, m_0, m_1, \{\mathsf{sk}_i\}_{i\in[q]})$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathsf{sk}_i \leftarrow \mathsf{FE.Keygen}(\mathsf{msk}, C_i)$ for all $i \in [q]$, and $b \leftarrow \{0,1\}$. When $q = 1$, we say that $\mathsf{FE}$ is selectively secure.*

**Definition 3.21** (Succinct Functional Encryption [22, 4, 75])**.** *An FE scheme* $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *for a class of circuits $\mathcal{C}^{s,n}$ with one-bit outputs is a* succinct *FE scheme if it holds that*

$$\mathsf{Time}\left[\mathsf{Enc}(\mathsf{pk}, m)\right] = \mathsf{poly}(\lambda, n(\lambda), \log(s(\lambda))),$$

*for every $\lambda \in \mathbb{N}$, $\mathsf{pk} \leftarrow \mathsf{Setup}(1^\lambda)$ and $m \in \{0,1\}^{n(\lambda)}$.*

We will use the succinct FE scheme of Goldwasser et al. [55] which is based on LWE (see [24, 75] for restatements). In particular, [55] gives a construction of succinct FE for $\mathsf{NC}^1$. This can be bootstrapped to a succinct FE scheme for $\mathsf{P}$ using a tranformation in [1], which additionally relies on the existence of symmetric-key encryption with decryption in $\mathsf{NC}^1$. Nevertheless, this can be based on LWE, which gives the following theorem.

**Theorem 3.22** ([55, 1])**.** *Assuming LWE, there exists a succinct FE scheme* $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *for any class of polynomial size circuits $\mathcal{C}^{s,n} = \{C_\lambda\}_\lambda$ with one-bit outputs. Moreover, it holds that*

$$\mathsf{Outlen}\left[\mathsf{Setup}(1^\lambda)\right] = \mathsf{poly}(\lambda, n, \log(s))$$
$$\mathsf{Outlen}\left[\mathsf{Keygen}(\mathsf{msk}, C)\right] = s \cdot \mathsf{poly}(\lambda, \log(s))$$

*for fixed polynomials, where $n = n(\lambda)$, $s = s(\lambda)$, $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $C \in C_\lambda$, and $m \in \{0,1\}^{n(\lambda)}$.*

Furthermore, we will need an FE scheme that supports functions with multiple output bits and satisfies a specific notion of efficiency. We obtain the following corollary by simple parallel repetition of the scheme of [54].

**Corollary 3.23** ([55, 57])**.** *Let* $\mathsf{sFE}$ *be a succinct FE scheme for the class of polynomial size circuits with one-bit outputs. Then, for every polynomial $q = q(\lambda)$, there exists a $q$-selectively-secure FE scheme* $\mathsf{IFE}$ *for circuits with $q$ output bits. The output of all algorithms is $q$ times that of* $\mathsf{sFE}$*. We call such a scheme a $q$-output succinct FE scheme.*

# 4 Correctness Amplification

In this section, we present a correctness amplification procedure for XiO. We show that assuming the existence of an XiO scheme with very weak correctness, there exists an XiO construction with a very strong correctness guarantee.

**Theorem 4.1.** *Let $p(\cdot)$ be any polynomial. Let $\mathsf{xi}\mathcal{O}$ be an XiO scheme for $\mathsf{P}^{\log}$ that is $\left(\frac{1}{2} + \frac{1}{p(\lambda)}\right)$-approximately correct. Assuming LWE and the existence of NIZKs, there exists a perfectly correct XiO scheme for $\mathsf{P}^{\log}$.*

The correctness amplification proceeds in three phases. First, in Section 4.1, we transform an approximately-correct XiO scheme to a $(1/\mathsf{poly}(\lambda) - \mathsf{negl}(\lambda))$-worst-case correct XiO scheme. Then, in Section 4.2, we transform the $(1/\mathsf{poly}(\lambda) - \mathsf{negl}(\lambda))$-worst-case XiO scheme to a $(1 - \mathsf{negl}(\lambda))$-worst-case correct XiO scheme. Then, in Section 4.3, we transform the $(1 - \mathsf{negl}(\lambda))$-worst-case correct XiO scheme to a perfectly correct XiO scheme. In Section 4.4, we prove Theorem 4.1 and conclude our correctness amplification.

### 4.1 From Approximately-Correct XiO to Worst-Case Correct XiO

Fix any class of circuits $\mathcal{C}^{s,n} \in \mathsf{P}^{\log}$. Throughout this section, we let $s = s(\lambda)$ and $n = n(\lambda)$. Our transformation relies on the following primitives as building blocks:

- $\mathsf{xi}\mathcal{O} = (\mathsf{xi}\mathcal{O}.\mathsf{Obf}, \mathsf{xi}\mathcal{O}.\mathsf{Eval})$ is a $(1/2 + \gamma)$-approximately correct XiO scheme for $\mathsf{P}^{\log}$, where $\gamma = 1/p(\lambda)$ for some polynomial $p$.
- $\mathsf{ECC}$ is a Reed-Solomon $\left(\frac{8 \cdot 2^{\frac{n}{d}}}{\gamma\lambda}, \frac{2^{\frac{n}{d}}}{\lambda}, \frac{8 \cdot 2^{\frac{n}{d}}}{\gamma\lambda} - \frac{2^{\frac{n}{d}}}{\lambda} + 1\right)_{2^\lambda}$ erasure correcting code that can correct up to a $(1 - \frac{\gamma}{8})$-fraction of erasures using the algorithm $\mathsf{ECC.Dec}$, where $|\mathsf{ECC}|$ is a polynomial of degree $d - 1$ in its input length. We assume that all inputs to $\mathsf{ECC}$ are padded to size $2^{\frac{n}{d}}$ bits. We note that the circuit $\mathsf{ECC}$ is in $\mathsf{NC}^1$. We let $\ell_1 = O(\log(\lambda)) + \frac{n}{d}$ be the length of the output of $\mathsf{ECC}$.
- $\mathsf{LDC}$ is a binary error-correcting code that is $(\frac{1}{2} - \frac{\gamma}{4}, \mathsf{poly})$-list decodable using the algorithm $\mathsf{LDC.Dec}$. We let $\ell_2 = O(\log(\lambda) + \log(s) + \log(n))$ be the output length of $\mathsf{LDC}$ when run on inputs of size $\mathsf{poly}(\lambda, s, n)$.
- $\mathsf{IFE} = (\mathsf{IFE.Setup}, \mathsf{IFE.Enc}, \mathsf{IFE.Keygen}, \mathsf{IFE.Dec})$ is a $\lambda$-output succinct FE scheme for the class $\mathcal{C}^{s',n'} \in \mathsf{P}$ where $s' = \left(s \cdot 2^{\frac{n}{d}}\right)^{d-1} \cdot \mathsf{poly}(\lambda)$ and $n' = s \cdot \mathsf{poly}(\lambda, n)$.
- $\mathsf{PRF} = (\mathsf{PRF.Key}, \mathsf{PRF.Punc}, \mathsf{PRF.Eval})$ is a puncturable PRF.
- $\mathsf{C} = (\mathsf{C.Commit}, \mathsf{C.Open})$ is a commitment scheme.
- $\mathsf{NIZK} = (\mathsf{NIZK.Gen}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ is a Multi-NIZK proof system for the NP language $L$ given by $L = \Big\{(\mathsf{ct}, i, \mathsf{com}_C, \mathsf{com}_0, \mathsf{pk}) :$ either

  1. $\exists r_0, r_1, C$ such that $\mathsf{ct}$ encrypts $(C, i)$ and $\mathsf{com}_C$ is a commitment to $C$, that is, $\mathsf{ct} = \mathsf{IFE.Enc}(\mathsf{pk}, (C, i);\ r_0) \ \wedge\ \mathsf{com}_C = \mathsf{C.Commit}(C, r_1)$, or
  2. $\exists r$ s.t. $\mathsf{com}_0 = \mathsf{C.Commit}(1, r)\Big\}$,

  We let $t = t(\lambda) = \mathsf{poly}(\lambda, s, n)$ denote the upper bound on the length of statements and witnesses in $L$ when instantiated with security parameter $\lambda$ (with parameters as used in the following scheme).

In what follows, we denote by $C_{x_1 \cdots x_k}$ the circuit $C$ with the first $k$ bits hardwired to $x_1 \cdots x_k$. We let $T$ denote a circuit in $\mathcal{C}^{s \cdot 2^{\frac{n}{d}}, s}$ that receives as input a circuit and outputs its truth table. The transformation is as follows.

**Worst-case correct XiO scheme $\mathsf{xi}\mathcal{O}'$:**

- $\widetilde{C} \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ :
  1. Sample $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{IFE.Setup}(1^\lambda)$.

2. Generate a key $\mathsf{sk}_{\mathcal{U}} \leftarrow \mathsf{IFE.Keygen}(\mathsf{msk}, \mathcal{U})$ for the circuit $\mathcal{U}$ such that

$$\mathcal{U}(D, i) = \mathsf{ECC}(T(D))[i],$$

for any input circuit $D$, where $\mathsf{ECC}(T(D))[i]$ denotes the $i$th block of $\lambda$ bits of $\mathsf{ECC}(T(D))$.

3. For every $x \in \{0,1\}^{n-\frac{n}{d}}$:
   (a) Sample $K_0^x, K_1^x \leftarrow \mathsf{PRF.Key}(1^\lambda)$, and $\sigma^x \leftarrow \mathsf{NIZK.Gen}(1^\lambda, 1^t)$.
   (b) Create commitments $\mathsf{com}_{C_x}^x = \mathsf{C.Commit}(C_x, r_0^x)$ to $C_x$ and $\mathsf{com}_0^x = \mathsf{C.Commit}(0, r_1^x)$ to 0 using randomness $r_0^x \leftarrow \{0,1\}^\lambda$ and $r_1^x \leftarrow \{0,1\}^\lambda$.
   (c) Generate the circuit $G^x = G^x[C_x, \mathsf{pk}, K_0^x, K_1^x, \mathsf{com}_{C_x}^x, \mathsf{com}_0^x, r_0^x, \sigma^x]$ such that on input $(i, j)$ does the following:
      i. Let $\mathsf{ct} \leftarrow \mathsf{IFE.Enc}(\mathsf{pk}, (C_x, i);\ \mathsf{PRF.Eval}(K_0^x, i))$.
      ii. Construct a NIZK proof $\pi = \mathsf{NIZK.P}(\sigma^x, v, w; \mathsf{PRF.Eval}(K_1, i))$ for the statement $v = (\mathsf{ct}, i, \mathsf{com}_{C_x}^x, \mathsf{com}_0^x, \mathsf{pk})$ using the witness $w = (C_x, \mathsf{PRF.Eval}(K_0^x, i), r_0^x)$.
      iii. Output the $j$th bit of $\mathsf{LDC}(\mathsf{ct}, \pi)$, denoted by $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
   (d) Let $\widetilde{G}^x \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G^x)$ and let $\widetilde{C}^x = (\widetilde{G}^x, \sigma^x, \mathsf{com}_{C_x}^x, \mathsf{com}_0^x)$.
4. Output $\widetilde{C} = \left(\left\{\widetilde{C}^x\right\}_{x \in \{0,1\}^{n-\frac{n}{d}}}, \mathsf{sk}_{\mathcal{U}}, \mathsf{pk}\right)$.

- $y' \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Eval}(\widetilde{C}, x)$ :

1. Let $x = x_1 || x_2$ where $|x_1| = n - \frac{n}{d}$.
2. For every $i \in [2^{\ell_1}]$:
   (a) For every $j \in [2^{\ell_2}]$, let $c_{ij} = \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{G}^{x_1}, (i, j))$.
   (b) Run $\mathsf{LDC.Dec}(c_{i1}c_{i2}\cdots c_{i2^{\ell_2}})$ to obtain a list of possible decodings, where the $k$th element of the list is $(\mathsf{ct}_i^k, \pi_i^k)$.
   (c) Let $k^\star$ be the first index $k$ such that $\mathsf{NIZK.V}(\sigma, v_i^k, \pi_i^k) = 1$ where $v_i^k = (\mathsf{ct}_i^k, i, \mathsf{com}_{C_{x_1}}^{x_1}, \mathsf{com}_0^{x_1}, \mathsf{pk})$. Set $\mathsf{ct}_i = \mathsf{ct}_i^{k^\star}$ if $k^\star$ exists and otherwise set $\mathsf{ct}_i = \perp$.
   (d) Run $y_i \leftarrow \mathsf{IFE.Dec}(\mathsf{sk}_{\mathcal{U}}, \mathsf{ct}_i)$.
3. If there are at least $\frac{\gamma}{8} \cdot 2^{\ell_1}$ indices $i$ for which $\mathsf{ct}_i \neq \perp$, let $y = y_1 y_2 \cdots y_{2^{\ell_1}}$ and run $\mathsf{ECC.Dec}(y)$ and output the element corresponding to $x_2$. Otherwise, output $\perp$.

**Theorem 4.2.** *Assume that* PRF *is a puncturable PRF,* IFE *is a selectively-secure $\lambda$-output succinct FE scheme for $C^{s',n'}$,* C *is a commitment scheme, and* NIZK *is a Multi-NIZK for $L$. Fix any class of circuits $\mathcal{C}^{s,n} \in \mathsf{P}^{\log}$. Let $p(\cdot)$ be any polynomial. Then, if* $\mathsf{xi}\mathcal{O}$ *is a $(1/2 + 1/p(\lambda))$-approximately-correct XiO scheme for $\mathsf{P}^{\log}$, then* $\mathsf{xi}\mathcal{O}'$ *is a $\left(\frac{1}{16p(\lambda)} - \mathsf{negl}(\lambda)\right)$-worst-case correct XiO scheme for $\mathcal{C}^{s,n}$, for a negligible function* negl.

**Proof.** Fix any class $\mathcal{C}^{s,n} = \{C_\lambda\}_{\lambda \in \mathbb{N}} \in \mathsf{P}^{\log}$ and let $\gamma = \frac{1}{p(\lambda)}$ such that $\mathsf{xi}\mathcal{O}$ is $\left(\frac{1}{2} + \gamma\right)$-approximately correct. Let $\mathsf{xi}\mathcal{O}'$ be the scheme resulting from the above transformation. Let $n = n(\lambda)$, $s = s(\lambda)$, $\ell_1 = \ell_1(\lambda)$, and $\ell_2 = \ell_2(\lambda)$. We show that $\mathsf{xi}\mathcal{O}'$ is a $(\frac{\gamma}{16} - \mathsf{negl}(\lambda))$-worst-case correct XiO for $\mathcal{C}^{s,n}$.

**Worst-Case Correctness.** To show worst-case correctness for $\mathsf{xi}\mathcal{O}'$, we want to show that there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and $x \in \{0,1\}^n$,

$$\Pr_{\mathsf{xi}\mathcal{O}'.\mathsf{Obf}}\left[\widetilde{C} \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C); \mathsf{xi}\mathcal{O}'.\mathsf{Eval}(\widetilde{C}, x) = C(x)\right] \geq \frac{\gamma}{2} - \mathsf{negl}(\lambda).$$

Towards this end, let $x = x_1||x_2$ with $|x_1| = n - \frac{n}{d}$. Let $R$ denote the random coins used by $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ which determine the part of the obfuscation used in the evaluation of $\widetilde{C}$ on $x$. In particular, this includes the randomness for generating the key pair $(\mathsf{pk}, \mathsf{msk})$, the functional key $\mathsf{sk}_{\mathcal{U}}$, and the values in $\widetilde{C}^{x_1}$.

Consider the $\mathsf{xi}\mathcal{O}'$ evaluation algorithm $\mathsf{xi}\mathcal{O}'.\mathsf{Eval}(\widetilde{C}, x)$. It first evaluates $\widetilde{G}^{x_1}$ to obtain $c_{ij} \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{G}^{x_1}, (i,j))$ for each $i \in [2^{\ell_1}]$ and $j \in [2^{\ell_2}]$. We start by an averaging argument which shows that conditioned on the choice of $R$, there is a noticeable size set of indices $i$ for which the majority of the evaluations $c_{ij}$ are correct. Define

$$S = \left\{ R : \Pr_{i,j}\left[\widetilde{G}^{x_1} \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G^{x_1}; R) : \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{G}^{x_1}, (i,j)) = G^{x_1}(i,j)\right] \geq \frac{1}{2} + \frac{\gamma}{2} \right\},$$

which will be the set of "good" random strings $R$, for which the obfuscated circuit, when obfuscated using randomness $R$, is correct on a majority of inputs. By an averaging argument and the $\left(\frac{1}{2}+\gamma\right)$-correctness of $\mathsf{xi}\mathcal{O}$, we have that a $\frac{\gamma}{2}$-fraction of the $R$ are in $S$. Let

$$S_R = \left\{ i : \Pr_{j}\left[\widetilde{G}^{x_1} \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G^{x_1}; R) : \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{G}^{x_1}, (i,j)) = G^{x_1}(i,j)\right] \geq \frac{1}{2} + \frac{\gamma}{4} \right\},$$

which will be the set of "good" inputs $i$, such that when $C$ is obfuscated using randomness $R$, for any $i \in S_R$, the resulting obfuscation is correct on inputs $(i,j)$ for a majority of the $j$. Then, for any $R \in S$, by an averaging argument, it holds that a $\frac{\gamma}{4}$-fraction of the inputs $i$ are in $S_R$.

We now show that for any $R \in S$ and $i \in S_R$, the evaluation algorithm $\mathsf{xi}\mathcal{O}'.\mathsf{Eval}$ obtains a valid $\mathsf{IFE}$ encryption of $(C_{x_1}, i)$. This is due to the guarantees of $\mathsf{LDC}$, $\mathsf{NIZK}$, $\mathsf{Commit}$, and $\mathsf{IFE}$. Fix $R \in S$ and $i \in S_R$.

After computing $c_{ij}$ for all $j$, the evaluation algorithm runs $\mathsf{LDC.Dec}(c_{i1}c_{i2}\ldots c_{i2^{\ell_2}})$. Since we are using a list-decoding algorithm, this results in a list of candidates of the form $(\mathsf{ct}_i^k, \pi_i^k)$ for polynomially many $k$. Recall that $\mathsf{LDC.Dec}$ is a $\left(\frac{1}{2} - \frac{\gamma}{4}, \mathsf{poly}\right)$-list-decoding algorithm, and can therefore correct a $\left(\frac{1}{2} - \frac{\gamma}{4}\right)$-fraction of errors. Since $R \in S$ and $i \in S_R$, we have that at most a $\left(\frac{1}{2} - \frac{\gamma}{4}\right)$-fraction of the $c_{ij}$ are incorrect, so the correct encryption of $(C_{x_1}, i)$ is in the decoded list. Let $k^\star$ be the index of the correct element.

To identify $k^\star$, we check each proof $\pi_i^k$. More concretely, for each $k$, let $v_i^k$ denote the statement $(\mathsf{ct}_i^k, i, \mathsf{com}_{C_{x_1}}^{x_1}, \mathsf{com}_0^{x_1}, \mathsf{pk})$, such that $\pi_i^k$ is a proof for the statement $v_i^k$. We note that $\mathsf{com}_{C_{x_1}}^{x_1}, \mathsf{com}_0^{x_1}$, and $\mathsf{pk}$ are part of the output of the obfuscation algorithm, and $i$ is used as input to obtain $(\mathsf{ct}_i^k, \pi_i^k)$, so the only unknown part of $v_i^k$ is $\mathsf{ct}_i^k$. Recall that the Multi-NIZK proof is for a language $L$ such that $(\mathsf{ct}, i, \mathsf{com}_{C_{x_1}}^{x_1}, \mathsf{com}_0^{x_1}, \mathsf{pk}) \in L$ if either

1. There exists a circuit $C'$ such that $\mathsf{ct}$ is an encryption of $(C', i)$ under $\mathsf{pk}$ and $\mathsf{com}_{C_{x_1}}^{x_1}$ is a commitment of $C'$. A witness for this statement consists of the randomness used to generate $\mathsf{ct}$, the opening for $\mathsf{com}_{C_{x_1}}^{x_1}$, and the circuit $C'$.

2. $\mathsf{com}_0^{x_1}$ is a commitment to 1. A witness for this statement is the opening for $\mathsf{com}_0^{x_1}$.

We now show that the decoding can correctly identify $k^\star$. For each $k$, the evaluation algorithm checks if $\mathsf{NIZK.V}(\sigma, v_i^k, \pi_i^k) = 1$. Since $k^\star$ is the index of the correct element, then $v_i^{k^\star} \in L$. As a result, by the completeness of $\mathsf{NIZK}$, the verification algorithm on $v_i^{k^\star}$ and $\pi_i^{k^\star}$ accepts, so at least one index $k$ passes the verification.

We now show that for any index $k$ that passes the verification, $\mathsf{ct}_i^k$ is an encryption of $(C_{x_1}, i)$. Suppose there exists $\hat{k}$ such that the verification algorithm accepts $\pi_i^{\hat{k}}$ for the statement $v_i^{\hat{k}}$. By the soundness of $\mathsf{NIZK}$, if $v_i^{\hat{k}} \notin L$ then the probability that the verification passes for $\pi_i^{\hat{k}}$ is negligible. Therefore, with overwhelming probability, $v_i^{\hat{k}} \in L$. Note that by the binding property of

the commitment scheme, there does not exist an $\hat{r}$ such that $\mathsf{com}_0^{x_1} = \mathsf{C.Commit}(1, \hat{r})$. Therefore, since $v_i^{\hat{k}} \in L$, it satisfies the first condition for being in $L$. Thus, by definition of $L$, it must be the case that there exists $\hat{r}_0, \hat{r}_1, \widehat{C}$ such that $\mathsf{ct}_i^{\hat{k}} = \mathsf{IFE.Enc}(\mathsf{pk}, (\widehat{C}, i); \hat{r}_0)$ and $\mathsf{com}_{C_{x_1}}^{x_1} = \mathsf{C.Commit}(\widehat{C}, \hat{r}_1)$. However, recall that $\mathsf{Commit}$ is binding, so $C_{x_1} = \widehat{C}$. Therefore, by the correctness of $\mathsf{IFE.Enc}$, any index $\hat{k}$ such that $\pi_i^{\hat{k}}$ is an accepting proof of $x_i^{\hat{k}}$ corresponds to an encryption of $(C_{x_1}, i)$. Therefore, for every $i$,

$$\Pr_R[\exists \hat{r} : \mathsf{ct}_i = \mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i); \hat{r}) \mid i \in S_R, R \in S] \geq 1 - \mathsf{negl}(\lambda) \tag{1}$$

for some negligible function $\mathsf{negl}$, which depends on the soundness of $\mathsf{NIZK}$.

Observe that even for $i \notin S_R$, any index $k$ that passes the verification corresponds to a correct encryption of $(C_{x_1}, i)$ by the argument above. Therefore, for any index $i$ such that $\mathsf{ct}_i \neq \bot$, we have that with high probability, $\mathsf{ct}_i$ is an encryption of $(C_{x_1}, i)$, that is,

$$\Pr_R[\exists \hat{r} : \mathsf{ct}_i = \mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i); \hat{r}) \mid \mathsf{ct}_i \neq \bot, R \in S] \geq 1 - \mathsf{negl}(\lambda).$$

After computing $\mathsf{ct}_i$ for each $i$, the evaluation algorithm runs $y_i \leftarrow \mathsf{IFE.Dec}(\mathsf{sk}_{\mathcal{U}}, \mathsf{ct}_i)$. By the correctness of $\mathsf{IFE.Dec}$ and by the argument above, there exists a negligible function $\mathsf{negl}$ such that for every $i$,

$$\Pr_R[y_i = \mathsf{ECC}(T(C_{x_1}))[i] \mid \mathsf{ct}_i \neq \bot, R \in S] \geq 1 - \mathsf{negl}(\lambda).$$

Let $I_R$ be the set of all indices $i$ for which $\mathsf{ct}_i \neq \bot$ when randomness $R$ is used. Since $|I_R| \leq \mathsf{poly}(\lambda)$, by a union bound we have that for some negligible function $\mathsf{negl}$,

$$\Pr_R[\forall i \in I_R : y_i = \mathsf{ECC}(T(C_{x_1}))[i] \mid R \in S] \geq 1 - \mathsf{negl}(\lambda). \tag{2}$$

To finish the proof, we show that the evaluation algorithm correctly computes $C(x)$ with probability $\frac{\gamma}{16} - \mathsf{negl}(\lambda)$. The evaluation algorithm proceeds by running $\mathsf{ECC.Dec}(y_1 y_2 \cdots y_{2^{\ell_1}})$, which can correct up to a $(1 - \frac{\gamma}{8})$-fraction of erasures. Because we know the indices $i$ for which we did not obtain $y_i$, this implies that $\mathsf{ECC.Dec}(y_1 y_2 \cdots y_{2^{\ell_1}}) = T(C_{x_1})$ if there at most a $(1 - \frac{\gamma}{8})$-fraction of symbols that have been erased, and all symbols that have not been erased are correct. We show that $I_R$ satisfies these requirements when $R \in S$. By Equation (1),

$$\begin{aligned}
\Pr_{R,i}[i \in I_R \mid R \in S] &\geq \Pr_{R,i}[i \in I_R \mid i \in S_R, R \in S] \cdot \Pr_{R,i}[i \in S_R \mid R \in S] \\
&\geq \Pr_{R,i}[\mathsf{ct}_i \neq \bot \mid i \in S_R, R \in S] \cdot \frac{\gamma}{4} \\
&\geq \Pr_{R,i}[\exists \hat{r} : \mathsf{ct}_i = \mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i); \hat{r}) \mid i \in S_R, R \in S] \cdot \frac{\gamma}{4} \\
&\geq \frac{\gamma}{4} - \mathsf{negl}(\lambda)
\end{aligned}$$

for a negligible function $\mathsf{negl}$. Therefore, by an averaging argument, we have that

$$\Pr_R\left[|I_R| \geq \frac{\gamma}{8} \cdot 2^{\ell_1} \;\middle|\; R \in S\right] \geq \frac{\gamma}{8} - \mathsf{negl}(\lambda). \tag{3}$$

Then, by a union bound, it follows from Equations (2) and (3) that

$$\Pr_R\left[\forall i \in I_R : y_i = \mathsf{ECC}(T(C_{x_1}))[i] \wedge |I_R| \geq \frac{\gamma}{8} \cdot 2^{\ell_1} \;\middle|\; R \in S\right] \geq \frac{\gamma}{8} - \mathsf{negl}(\lambda)$$

for a negligible function $\mathsf{negl}$. Let $y'$ be the element of $\mathsf{ECC.Dec}(y_1 y_2 \cdots y_{2^{\ell_1}})$ corresponding to $x_2$. We have that

$$\begin{aligned}
\Pr_R\left[y' = C(x)\right] &\geq \Pr_R\left[\mathsf{ECC.Dec}(y_1 y_2 \cdots y_{2^{\ell_1}}) = T(C_{x_1})\right] \\
&\geq \Pr_R\left[\mathsf{ECC.Dec}(y_1 y_2 \cdots y_{2^{\ell_1}}) = T(C_{x_1}) \mid R \in S\right] \cdot \Pr_R\left[R \in S\right] \\
&\geq \Pr_R\left[\forall i \in I_R : y_i = \mathsf{ECC}(T(C_{x_1}))[i] \wedge |I_R| \geq \frac{2^{\ell_1} \cdot \gamma}{8} \;\middle|\; R \in S\right] \cdot \Pr_R\left[R \in S\right] \\
&\geq \left(\frac{\gamma}{8} - \mathsf{negl}(\lambda)\right) \cdot \frac{\gamma}{2} \geq \frac{\gamma}{16} - \mathsf{negl}(\lambda)
\end{aligned}$$

for some negligible function $\mathsf{negl}$, as desired.

**Compression.** Fix any $C \in \mathcal{C}_\lambda$ and $x \in \{0,1\}^n$, and let $x = x_1 || x_2$ with $|x_1| = n - \frac{n}{d}$. We first bound the size of the circuit $G^{x_1}$ obfuscated during $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$. We have that $G^{x_1}(i,j)$ computes an $\mathsf{IFE}$ encryption $\mathsf{ct}$, a $\mathsf{NIZK}$ proof $\pi$, an $\mathsf{LDC}$ encoding, and evaluations of $\mathsf{PRF}$. The $\mathsf{LDC}$ encoding and $\mathsf{PRF}$ run in time polynomial in their input, thus we only have to analyze $|\mathsf{ct}|$ and $|\pi|$. Observe that $\mathsf{IFE}$ is used to generate a key for the circuit $\mathcal{U}$, which is contained in $\mathcal{C}^{s',n'}$ where $s' \leq s^d \cdot 2^n \cdot \mathsf{poly}(\lambda)$ and $n' \leq s \cdot \mathsf{poly}(\lambda, n)$. Therefore, we have that

$$|\mathsf{ct}| = |\mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i))| = \lambda \cdot \mathsf{poly}(\lambda, |C_{x_1}| + |i|, \log(s')) \leq \mathsf{poly}(\lambda, s, n, \log(2^n)) = \mathsf{poly}(\lambda, s, n)$$

by the $\lambda$-output succinctness of $\mathsf{IFE}$ and

$$\begin{aligned}
|\pi| &= \mathsf{poly}\left(|\mathsf{ct}|, |i|, \left|\mathsf{com}^{x_1}_{C_{x_1}}\right|, |\mathsf{com}^{x_1}_0|, |\mathsf{pk}|\right) + \mathsf{poly}\left(|C_{x_1}|, |\mathsf{PRF.Eval}(K^{x_1}_0, i)|, \left|\mathsf{com}^{x_1}_{C_{x_1}}\right|\right) \\
&= \mathsf{poly}(\lambda, s, n),
\end{aligned}$$

because $\mathsf{NIZK.P}$ is polynomial in the length of the statements and witnesses for $L$, where we used the fact that $|\mathsf{pk}|$ is polynomial in the depth of $\mathcal{U}$ and input length. This follows because $\mathsf{ECC}$ can be computed in $\mathsf{NC}^1$ and $T$ has depth $\mathsf{poly}(s)$, so the depth of $\mathcal{U}$ is $s + \log(|T(C)| \cdot \mathsf{poly}(\lambda) = \mathsf{poly}(\lambda, s, n)$. Therefore, we have that $|G^{x_1}| \leq \mathsf{poly}(\lambda, s, n)$.

To bound the efficiency of $\mathsf{xi}\mathcal{O}'$, we have that $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ runs $\mathsf{PRF.Key}$, $\mathsf{NIZK.Gen}$, $\mathsf{C.Commit}$ for each $x \in \{0,1\}^{n - \frac{n}{d}}$, which all have time bounded by $\mathsf{poly}(\lambda, s, n)$, as well as $\mathsf{IFE.Setup}$, $\mathsf{IFE.Keygen}$, and $2^{n-\frac{n}{d}}$ instances of $\mathsf{xi}\mathcal{O}.\mathsf{Obf}$. Therefore, for every $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_\lambda$, and $x \in \{0,1\}^n$ we have that by the compression of $\mathsf{xi}\mathcal{O}$ and $\lambda$-output succinctness of $\mathsf{IFE}$,

$$\begin{aligned}
&\mathsf{Time}\left[\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)\right] \\
&= 2^{n-\frac{n}{d}} \cdot \left(\mathsf{poly}(\lambda, s, n) + \mathsf{Time}\left[\mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G^{x_1})\right]\right) + \mathsf{Time}\left[\mathsf{IFE.Setup}(1^\lambda)\right] + \mathsf{Time}\left[\mathsf{IFE.Keygen}(\mathsf{msk}, \mathcal{U})\right] \\
&\leq \mathsf{poly}(\lambda, s, 2^n) + 2^{n-\frac{n}{d}} \cdot \mathsf{poly}\left(\lambda, |G^{x_1}|, 2^{\ell_1 + \ell_2}\right) + \mathsf{poly}(\lambda, s, n, \log(s^d \cdot 2^n)) + \mathsf{poly}(\lambda, s^d, 2^n) \\
&\leq \mathsf{poly}(\lambda, s, 2^n) + 2^{n-\frac{n}{d}} \cdot \mathsf{poly}\left(\lambda, s, n, 2^{O(\log(\lambda) + \log(s) + \log(n)) + \frac{n}{d}}\right) + \mathsf{poly}(\lambda, s, n) + \mathsf{poly}(\lambda, s, 2^n) \\
&\leq \mathsf{poly}(\lambda, s, 2^n)
\end{aligned}$$

and

$$\begin{aligned}
&\mathsf{Outlen}\left[\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)\right] = |\mathsf{sk}_\mathcal{U}| + |\mathsf{pk}| + 2^{n-\frac{n}{d}}\left(2\left|\mathsf{com}^{x_1}_{C_{x_1}}\right| + |\sigma^{x_1}| + \mathsf{Outlen}\left[\mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G^{x_1})\right]\right) \\
&\leq \left(s \cdot 2^{\frac{n}{d}}\right)^{d-1} \cdot \mathsf{poly}(\lambda, s, n) + \mathsf{poly}(\lambda, s, n) + 2^{n-\frac{n}{d}}\left(\mathsf{poly}(\lambda, s) + \mathsf{poly}(\lambda, s, n) + \mathsf{poly}(\lambda, |G^{x_1}|) \cdot 2^{(\ell_1+\ell_2)(1-\epsilon)}\right) \\
&\leq \left(2^{n\cdot\left(1-\frac{1}{d}\right)} + 2^{n\left(1-\frac{1}{d}\right)+\frac{n}{d}(1-\epsilon)}\right) \cdot \mathsf{poly}(\lambda, s, n) \leq 2^{n\left(1-\frac{\epsilon}{d}\right)} \cdot \mathsf{poly}(\lambda, s)
\end{aligned}$$

for a constant $\epsilon$ which depends on the efficiency of $\mathsf{xi}\mathcal{O}$, where we used the fact that $\ell_1 = O(\log(\lambda)) + \log(s) + \frac{n}{d}$ and $\ell_2 = O(\log(\lambda) + \log(s) + \log(n))$ and Theorem 3.22 and Corollary 3.23.

**Indistinguishability.** We now turn to the security of $\mathsf{xi}\mathcal{O}'$. Let $C^0, C^1 \in \mathcal{C}_\lambda$ be functionally equivalent. We show that for all PPT adversaries $\mathcal{A}$, the probability of outputting $b$ on input $(C^0, C^1, \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b))$ is at most negligibly far from $\frac{1}{2}$ where $b \leftarrow \{0,1\}$.

To do this, we first show that for each $x \in \{0,1\}^{n-\frac{n}{d}}$, the probability of outputting $b$ on input $(C^0, C^1, \mathsf{pk}, \mathsf{sk}_\mathcal{U}, \widetilde{C}^x)$ is at most negligibly far from $\frac{1}{2}$ where $b \leftarrow \{0,1\}$, and $\mathsf{pk}, \mathsf{sk}_\mathcal{U}, \widetilde{C}^x$ are as in $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$. To formalize this, let $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$ denote the steps of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}$ which result in $\widetilde{C}^x$. In particular, $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$ does the following:

1. Sample $K_0, K_1 \leftarrow \mathsf{PRF.Key}(1^\lambda)$, and $\sigma \leftarrow \mathsf{NIZK.Gen}(1^\lambda, 1^t)$.
2. Create commitments $\mathsf{com}_{C_x} = \mathsf{C.Commit}(C_x, r_0)$ to $C_x$ and $\mathsf{com}_0 = \mathsf{C.Commit}(0, r_1)$ to 0 using randomness $r_0 \leftarrow \{0,1\}^\lambda$ and $r_1 \leftarrow \{0,1\}^\lambda$.
3. Generate the circuit $G = G[C_x, \mathsf{pk}, K_0, K_1, \mathsf{com}_{C_x}, \mathsf{com}_0, r_0, \sigma]$ such that on input $(i, j)$ does the following:
   (a) Let $\mathsf{ct} \leftarrow \mathsf{IFE.Enc}(\mathsf{pk}, (C_x, i);\ \mathsf{PRF.Eval}(K_0, i))$.
   (b) Construct a NIZK proof $\pi = \mathsf{NIZK.P}(1^\lambda, \sigma, v, w; \mathsf{PRF.Eval}(K_1, i))$ for the statement $v = (\mathsf{ct}, i, \mathsf{com}_{C_x}, \mathsf{com}_0, \mathsf{pk})$ using the witness $w = (C_x, \mathsf{PRF.Eval}(K_0, i), r_0)$.
   (c) Output the $j$th bit of $\mathsf{LDC}(\mathsf{ct}, \pi)$, denoted by $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
4. Let $\widetilde{G} \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, G)$ and output $\widetilde{C}^x = (\widetilde{G}, \mathsf{com}_{C_x}, \mathsf{com}_0, \sigma)$.

Then, we can write $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ as follows:

1. Sample $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{IFE.Setup}(1^\lambda)$.
2. Generate a key $\mathsf{sk}_\mathcal{U} \leftarrow \mathsf{IFE.Keygen}(\mathsf{msk}, \mathcal{U})$ for $\mathcal{U}$, where $\mathcal{U}(C_x, i) = \mathsf{ECC}(T(C))[i]$.
3. For every $x \in \{0,1\}^{n-\frac{n}{d}}$, run $\widetilde{C}^x \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$.
4. Output $\widetilde{C} = \left(\{\widetilde{C}^x\}_{x \in \{0,1\}^{n-\frac{n}{d}}}, \mathsf{sk}_\mathcal{U}, \mathsf{pk}\right)$.

We use this formulation and notation to prove security of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}$ (in particular, for ease of notation, we omit the superscript $x$ from the values $K_0, K_1, \mathsf{com}_{C_x}, \mathsf{com}_0, \sigma, r_0, r_1$ and $G$ used to generate $\widetilde{C}^x$ when it is clear from context). We first show that for each $x$, the probability of outputting $b$ on input $(C^0, C^1, \mathsf{pk}, \mathsf{sk}_\mathcal{U}, \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x])$ is at most negligibly far from $\frac{1}{2}$ where $b \leftarrow \{0,1\}$. For this part of the proof, for each $x \in \{0,1\}^{n-\frac{n}{d}}$ we have the following hybrids:

**Phase I: Changing the commitment $\mathsf{com}_0$.** We begin with the real execution, where $\mathsf{pk}$ and $\mathsf{sk}_\mathcal{U}$, and $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$ are generated honestly. In particular, $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$ uses $\mathsf{xi}\mathcal{O}$ to obfuscate the circuit $G$ such that on input $(i, j)$, generates a ciphertext $\mathsf{ct}$ of $(C_x^b, i)$ and a proof $\pi$ that $(\mathsf{ct}, i, \mathsf{com}_{C_x^b}, \mathsf{com}_0, \mathsf{pk})$ satisfies the first condition for being in $L$. Since the opening to $\mathsf{com}_0$ is not used, we can change $\mathsf{com}_0$ to be a commitment $\mathsf{com}_1$ to 1, relying on the hiding property of the commitment scheme.

**Phase II: Switching the witness for the Multi-NIZK proof.** We then go through a series of hybrids to switch the witness for the proof $\pi$ generated by $G$ to be a witness for the second condition to being in $L$. This relies on the witness indistinguishability of the Multi-NIZK, which follows from the ZK property.

**Phase III: Switching the commitment of $C_x^b$.** Because the Multi-NIZK $\pi$ generated by $G$ now uses a witness to the second condition for being in $L$, the opening to $\mathsf{com}_{C_x^b}$ no longer needs to appear in the circuit $G$. Therefore, we can change $\mathsf{com}_{C_x^b}$ to a commitment $\mathsf{com}_{C_x^0}$ of $C_x^0$. Indistinguishability follows from the hiding property of the commitment scheme.

**Phase IV: Switching the encryption.** We then go through a sequence of hybrids to switch $\mathsf{ct}$ from an encryption of $(C_x^b, i)$ to an encryption of $(C_x^0, i)$. After this change, the output of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x]$ is independent of $b$.

Fix any $x \in \{0,1\}^{n-\frac{n}{d}}$. We now present the formal description of each hybrid.

**Phase I: Changing the commitments $\mathsf{com}_0$.**

- **$\mathsf{Hyb}^1(\lambda)$:** In this hybrid, we first sample $(\mathsf{msk}, \mathsf{pk}) \leftarrow \mathsf{IFE.Setup}(1^\lambda)$ and $\mathsf{sk}_{\mathcal{U}} \leftarrow \mathsf{IFE.Keygen}(\mathsf{msk}, \mathcal{U})$ as in the real execution of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)$. Then, we run $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[x, \mathsf{pk}]$ as in the real execution, as follows:

  1. Sample $K_0, K_1 \leftarrow \mathsf{PRF.Key}(1^\lambda)$, and $\sigma \leftarrow \mathsf{NIZK.Gen}(1^\lambda, 1^t)$.
  2. Create commitments $\mathsf{com}_{C_x^b} = \mathsf{C.Commit}(C_x^b, r_0)$ and $\mathsf{com}_0 = \mathsf{C.Commit}(0, r_1)$ using randomness $r_0, r_1$.
  3. Obfuscate the following circuit $G^1 = G^1[C_x^b, \mathsf{pk}, K_0, K_1, \mathsf{com}_{C_x^b}, \mathsf{com}_0, r_0, \sigma]$ to obtain $\widetilde{G}$, such that $G^1(i,j)$ does the following:
     (a) $\mathsf{ct} \leftarrow \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^b, i); \mathsf{PRF.Eval}(K_0, i)\right)$.
     (b) $\pi = \mathsf{NIZK.P}\left(\sigma, v, \left(C_x^b, \mathsf{PRF.Eval}(K_0, i), r_0\right); \hat{r}\right)$ where $v = (\mathsf{ct}, i, \mathsf{com}_{C_x^b}, \mathsf{com}_0, \mathsf{pk})$ and $\hat{r} = \mathsf{PRF.Eval}\left(K_1, i\right)$.
     (c) Output $\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
  4. Output $\widetilde{C}^x = (\widetilde{G}, \mathsf{com}_{C_x^b}, \mathsf{com}_0, \sigma)$.

  The output of this experiment is $(\mathsf{pk}, \mathsf{sk}_{\mathcal{U}}, \widetilde{C}^x)$.

- **$\mathsf{Hyb}^2(\lambda)$:** This experiment is obtained from the previous experiment by replacing $\mathsf{com}_0$ with $\mathsf{com}_1 = \mathsf{C.Commit}(1, r_0)$.

  This is indistinguishable from the previous hybrid by the hiding property of the commitment scheme.

**Phase II: Switching the witness for the Multi-NIZK proof.**

- **$\mathsf{Hyb}_z^{3.1}(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by modifying $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x]$ as follows:

  1. Generate $K_0, K_1, \sigma, r_0, r_1, \mathsf{com}_{C_x^b}$, and $\mathsf{com}_1$ as in the previous hybrid.
  2. Puncture $K_1$ to obtain $\widetilde{K}_1^{[z]} \leftarrow \mathsf{PRF.Punc}(K_1, z)$.
  3. Set $r^\star = \mathsf{PRF.Eval}(K_1, z)$.
  4. Set $\mathsf{ct}^\star = \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^b, z); \mathsf{PRF.Eval}(K_0, z)\right)$.
  5. Set $\pi^\star = \mathsf{NIZK.P}(\sigma, v, w; r^\star)$, where the statement $v = (\mathsf{ct}^\star, z, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk})$ and the witness $w = (C_x^b, \mathsf{PRF.Eval}(K_0, z), r_0)$.
  6. Obfuscate the following circuit $G_z^{3.1} = G_z^{3.1}[C_x^b, \mathsf{pk}, K_0, \widetilde{K}_1^{[z]}, \mathsf{com}_{C_x^b}, \mathsf{com}_1, r_0, r_1, \sigma, \pi^\star]$ to obtain $\widetilde{G}$, such that $G_z^{3.1}(i,j)$ does the following:
     (a) $\mathsf{ct} \leftarrow \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^b, i); \mathsf{PRF.Eval}(K_0, i)\right)$.

(b) $\pi = \begin{cases} \mathsf{NIZK.P}\left(\sigma, v, r_1; \hat{r}\right) & \text{if } i < z \\ \pi^\star & \text{if } i = z \\ \mathsf{NIZK.P}\left(\sigma, v, \left(C^b_x, \mathsf{PRF.Eval}(K_0, i), r_0\right); \hat{r}\right) & \text{if } i > z \end{cases}$

where $v = (\mathsf{ct}, i, \mathsf{com}_{C^b_x}, \mathsf{com}_1, \mathsf{pk})$ and $\hat{r} = \mathsf{PRF.Eval}\left(\widetilde{K}_1^{[z]}, i\right)$. That is, if $i \geq z$, $\pi$ uses a witness for the first condition to being in $L$, and if $i < z$, then $\pi$ uses a witness for the second condition to being in $L$.

(c) Output $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.

7. Output $\widetilde{C}^x = (\widetilde{G}, \mathsf{com}_{C^b_x}, \mathsf{com}_0, \sigma)$.

For $z = 0^{\ell_1}$, we will show that $\mathsf{Hyb}^2(\lambda)$ is computationally indistinguishable from $\mathsf{Hyb}^{3.1}_z(\lambda)$, and for $z > 0^{\ell_1}$, we will show that $\mathsf{Hyb}^{3.4}_z(\lambda)$ and $\mathsf{Hyb}^{3.1}_{z+1}(\lambda)$ are computationally indistinguishable. Both of these proofs are due to the fact that the $\widetilde{G}$ generated in the corresponding hybrids are obfuscations of functionally equivalent circuits.

- **$\mathsf{Hyb}^{3.2}_z(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by replacing $r^\star$ with a truly random value.

  This experiment is indistinguishable from the previous hybrid because the output of the PRF is pseudorandom at punctured points.

- **$\mathsf{Hyb}^{3.3}_z(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by letting $\pi^\star = \mathsf{NIZK.P}(\sigma, (\mathsf{ct}^\star, z, \mathsf{com}_{C^b_x}, \mathsf{com}_1, \mathsf{pk}), r_1; r^\star)$, that is, $\pi^\star$ is now generated using a witness to the second condition for being in $L$.

  This is indistinguishable from the previous hybrid due to the witness indistinguishability of $\mathsf{NIZK}$.

- **$\mathsf{Hyb}^{3.4}_z(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by changing $r^\star$ to $\mathsf{PRF.Eval}(K_1, z)$.

  This is indistinguishable from the previous hybrid because the output of the PRF is pseudorandom at punctured points.

- **$\mathsf{Hyb}^4(\lambda)$:** This experiment is obtained from the previous experiment by changing $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x]$ to do the following:

  1. Generate $K_0, K_1, \sigma, r_0, r_1, \mathsf{com}_{C^b_x}$, and $\mathsf{com}_1$ as in the previous hybrid.
  2. Obfuscate the circuit $G^4 = G^4[C^b_x, \mathsf{pk}, K_0, K_1, \mathsf{com}_1, r_1, \sigma]$ to obtain $\widetilde{G}$ such that $G^4(i,j)$ does the following:
     (a) $\mathsf{ct} \leftarrow \mathsf{IFE.Enc}(\mathsf{pk}, (C^b_x, i); \mathsf{PRF.Eval}(K_0, i))$.
     (b) $\pi = \mathsf{NIZK.P}(\sigma, (\mathsf{ct}, i, \mathsf{com}_{C^b_x}, \mathsf{com}_1, \mathsf{pk}), r_1;\ \mathsf{PRF.Eval}(K_1, i))$, that is, $\pi$ is always a proof using $r_1$ as the witness to the second condition for being in $L$.
     (c) Output $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
  3. Output $\widetilde{C}^x = (\widetilde{G}, \mathsf{com}_{C^b_x}, \mathsf{com}_1, \sigma)$.

  This is indistinguishable from the previous hybrid $\mathsf{Hyb}^{3.4}_{1^{\ell_1}}(\lambda)$ since the circuits generated in both hybrids are functionally equivalent.

**Phase II: Switching the commitment $\mathsf{com}_{C_x^b}$.**

- **$\mathsf{Hyb}^5(\lambda)$:** This experiment is obtained from the previous experiment by changing the commitment $\mathsf{com}_{C_x^b}$ to $\mathsf{com}_{C_x^0} = \mathsf{Commit}(C_x^0, r_0)$.

  This is indistinguishable from the previous hybrid due to the hiding property of the commitment scheme.

**Phase III: Switching the encryption.**

- **$\mathsf{Hyb}_z^{6.1}(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by changing $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x]$ to do the following:

  1. Generate $K_0, K_1, \sigma, r_0, r_1, \mathsf{com}_{C_x^0}$ and $\mathsf{com}_1$ as in the previous hybrid.
  2. Puncture $K_0$ to obtain $\widetilde{K}_0^{[z]} \leftarrow \mathsf{PRF.Punc}(K_0, z)$.
  3. Set $r^\star = \mathsf{PRF.Eval}(K_0, z)$.
  4. Set $\mathsf{ct}^\star = \mathsf{IFE.Enc}(\mathsf{pk}, (C^b, z); r^\star)$.
  5. Obfuscate the following circuit $G_z^{6.1} = G_z^{6.1}[C_x^0, C_x^b, \mathsf{pk}, \widetilde{K}_0^{[z]}, K_1, \mathsf{com}_{C_x^0}, \mathsf{com}_1, r_1, \sigma, c^\star]$ to obtain $\widetilde{G}$ such that $G_z^{6.1}(i,j)$ does the following:
     (a) $\mathsf{ct} = \begin{cases} \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^0, i);\ \mathsf{PRF.Eval}\left(\widetilde{K}_0^{[z]}, i\right)\right) & \text{if } i < z \\ \mathsf{ct}^\star & \text{if } i = z. \\ \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^b, i);\ \mathsf{PRF.Eval}\left(\widetilde{K}_0^{[z]}, i\right)\right) & \text{if } i > z \end{cases}$
     (b) $\pi \leftarrow \mathsf{NIZK.P}(\sigma, (\mathsf{ct}, i, \mathsf{com}_{C_x^0}, \mathsf{com}_1, \mathsf{pk}), r_1;\ \mathsf{PRF.Eval}(K_1, 1))$.
     (c) Output $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
  6. Output $(\widetilde{G}, \mathsf{com}_{C_x^0}, \mathsf{com}_1, \sigma)$.

  We will show that $\mathsf{Hyb}_{0^{\ell_1}}^{6.1}(\lambda)$ and $\mathsf{Hyb}^5(\lambda)$ are computationally indistinguishable and that $\mathsf{Hyb}_{z+1}^{6.1}(\lambda)$ and $\mathsf{Hyb}_z^{6.4}(\lambda)$ are computationally indistinguishable for all $z \geq 0^{\ell_1}$. These hold because the obfuscated circuits are functionally equivalent.

- **$\mathsf{Hyb}_z^{6.2}(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by replacing $r^\star$ with a truly random value.

  This is indistinguishable from the previous hybrid because the output of the PRF is pseudorandom at punctured points.

- **$\mathsf{Hyb}_z^{6.3}(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by generating the hardcoded ciphertext as $\mathsf{ct}^\star = \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^0, z); r^\star)$, that is, $\mathsf{ct}^\star$ is now an encryption of $C_x^0$.

  This is indistinguishable from the previous hybrid because of the semantic security of $\mathsf{IFE}$.

- **$\mathsf{Hyb}_z^{6.4}(\lambda)$ for $z \in \{0,1\}^{\ell_1}$:** This experiment is obtained from the previous experiment by calculating $r^\star$ as $\mathsf{PRF.Eval}(K_0, z)$.

  This is indistinguishable form the previous experiment because the output of the PRF is pseudorandom at punctured points.

- **$\mathsf{Hyb}^7(\lambda)$:** This experiment is obtained from the previous experiment by changing $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x]$ to do the following:

  1. Generate $K_0, K_1, \sigma, r_0, r_1, \mathsf{com}_{C_x^0}$, and $\mathsf{com}_1$ as in the previous hybrid.

2. Obfuscate the following circuit $G^7 = G^7[C_x^0, \mathsf{pk}, K_0, K_1, \mathsf{com}_{C_x^0}, \mathsf{com}_1, r_1, \sigma]$ to obtain $\widetilde{G}$ such that $G^7(i,j)$ does the following:
   (a) $\mathsf{ct} = \mathsf{IFE.Enc}\left(\mathsf{pk}, (C_x^0, i);\ \mathsf{PRF.Eval}(K_0, i)\right)$.
   (b) $\pi = \mathsf{NIZK.P}(\sigma, (\mathsf{ct}, i, \mathsf{com}_{C_x^0}, \mathsf{com}_1, \mathsf{pk}), r_1;\ \mathsf{PRF.Eval}(K_1, i))$.
   (c) Output $(\mathsf{LDC}(\mathsf{ct}, \pi))_j$.
3. Output $(\widetilde{G}, \mathsf{com}_{C_x^0}, \mathsf{com}_1, \sigma)$.

This is indistinguishable from the previous hybrid because the circuit $G^7$ is functionally equivalent to the circuit $G^{6.4}_{1^{\ell_1}}$. Observe that at this point, the output of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b)[\mathsf{pk}, x]$ is independent of $b$, and therefore, no adversary can guess $b$ in this experiment with probability noticeably far from $\frac{1}{2}$.

We proceed by showing that each consecutive pair of hybrid experiments is computationally indistinguishable.

**Claim 4.3.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}^1(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^2(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** The difference between these two hybrids is that in $\mathsf{Hyb}^1(\lambda)$, $\mathsf{com}_0$ is a commitment to 0 and in $\mathsf{Hyb}^2(\lambda)$, it is replaced with $\mathsf{com}_1$, a commitment to 1. Note that the opening $r_1$ to $\mathsf{com}_0$ and $\mathsf{com}_1$ is not included in the circuits $G^1$ or $G^2$. Therefore, these hybrids are computationally indistinguishable by the hiding property of the commitment scheme. □

**Claim 4.4.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}^2(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^{3.1}_{0^{\ell_1}}(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** The difference between these two hybrids is that in $\mathsf{Hyb}^2(\lambda)$, $\widetilde{G}$ is an obfuscation of $G^2$ and in $\mathsf{Hyb}^{3.1}_{0^{\ell_1}}(\lambda)$, $\widetilde{G}$ is an obfuscation of $G^{3.1}_{0^{\ell_1}}$. We show that these two circuits are functionally equivalent.

When $i \neq 0^{\ell_1}$, the difference between the two circuits is that $G^2(i,j)$ uses the PRF key $K_1$ to generate the proof $\pi$ and $G^{3.1}_{0^{\ell_1}}(i,j)$ uses the punctured PRF key $\widetilde{K}_1^{[z]}$ to generate $\pi$. Since neither circuit evaluates the PRF on the punctured point when $i \neq 0^{\ell_1}$, the outputs of the two circuits are the same.

For $i = 0^{\ell_1}$, we have that the ciphertext $\mathsf{ct}$ generated by both circuits are the same, because both are computed with the unpunctured PRF key $K_0$. For the Multi-NIZK proof, the circuit $G^{3.1}_{0^{\ell_1}}$ has a hardcoded proof $\pi^\star$ which is calculated using the ciphertext $\mathsf{ct}^\star$. Since $\mathsf{ct}^\star$ is exactly the ciphertext generated by $G^2(0^{\ell_1}, j)$, then it holds that $\pi^\star$ is generated exactly as the proof $\pi$ in $G^2(0^{\ell_1}, j)$.

Therefore, the obfuscations of $G^2$ and $G^{3.1}_{0^{\ell_1}}$ are computationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

**Claim 4.5.** *For all $z \in \{0,1\}^{\ell_1}$, for any PPT adversary $\mathcal{A}$, it holds that* $\Big|\Pr\left[\mathcal{A}(\mathsf{Hyb}^{3.1}_z(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^{3.2}_z(\lambda)) = 1\right]\Big| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the pseudorandom property at punctured points. The difference between these hybrids is that on input $(z,j)$, the circuit $G^{3.1}_z$ computes $\pi$ as a hardcoded proof $\pi^\star$ calculated using $\mathsf{PRF.Eval}(K_1, z)$ as the randomness, and $G^{3.2}_z$ computes $\pi^\star$ using a truly random value for the randomness of the proof. If there was an adversary $\mathcal{A}$ that could distinguish between these two hybrids, then one could construct an adversary $\mathcal{B}$ that receives as input the punctured key $\widetilde{K}_1^{[z]}$, and a challenge $\hat{r}$ which is either $\mathsf{PRF.Eval}(K_1, z)$ or a uniformly random value. Then, $\mathcal{B}$ could sample $\mathsf{pk}$ and $\mathsf{sk}_\mathcal{U}$ honestly, and simulate generating $\widetilde{C}^x$ for $\mathcal{A}$ as in $\mathsf{Hyb}^{3.1}_z(\lambda)$, with the exception that $\mathcal{B}$ would set $r^\star = \hat{r}$ as the randomness for the hardcoded proof $\pi^\star$. Then, the distinguishing advantage of $\mathcal{A}$ would directly translate into the distinguishing advantage for $\mathcal{B}$. Therefore, these hybrids are computationally indistinguishably by the security of the PRF. □

**Claim 4.6.** *For all $z \in \{0,1\}^{\ell_1}$, For any PPT adversary $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{3.2}(\lambda))=1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{3.3}(\lambda))=1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the witness indistinguishability of the Multi-NIZK. The difference between these hybrids is that in $\mathsf{Hyb}_z^{3.2}(\lambda)$, $\pi^\star$ is a proof that $(\mathsf{ct}^\star, z, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk}) \in L$ using a witness to the first condition for being in $L$, while in $\mathsf{Hyb}_z^{3.3}(\lambda)$, $\pi^\star$ is a proof for the same statement using a witness to the second condition for being in $L$. Indistinguishability between adjacent hybrids follows from the witness indistinguishability of the Multi-NIZK. □

**Claim 4.7.** *For all $z \in \{0,1\}^{\ell_1}$, for any PPT adversary $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{3.3}(\lambda))=1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{3.4}(\lambda))=1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.**

This holds due to the pseudorandom property at punctured points. The difference between these hybrids is that on input $(z, j)$, the circuit $G_z^{3.3}$ computes $\pi$ as a hardcoded proof $\pi^\star$ calculated using a truly random value $r^\star$ as the randomness for the proof, and $G_z^{3.4}$ computes $\pi^\star$ using $r^\star = \mathsf{PRF.Eval}(K_1, z)$ for the randomness. Therefore, if there were an adversary that could distinguish between these hybrids, one could construct an adversary that would embed a PRF challenge as $r^\star$ and thus break the security of the PRF. □

**Claim 4.8.** *For every $z \in \{0,1\}^{\ell_1} \setminus \{1^{\ell_1}\}$, for any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{3.4}(\lambda))=1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_{z+1}^{3.1}(\lambda))=1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the security of $\mathsf{xi}\mathcal{O}$. Specifically, the difference between these two circuits is the way that the Multi-NIZK proofs are calculated in the circuits $G_{z+1}^{3.1}$ and $G_z^{3.4}$. When $i \notin \{z, z+1\}$, the computation done by the circuits is identical.

When $i = z$, the circuit $G_z^{3.4}(z, j)$ calculated $\pi$ as a hardcoded proof $\pi^\star$ for the statement $v = (\mathsf{ct}^\star, z, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk})$ using the witness $w = r_1$, and using randomness $\mathsf{PRF.Eval}(K_1, z)$. Similarly, $G_{z+1}^{3.1}(z, j)$ generates $\pi$ for the same statement $v$ and witness $w$, using randomness $\mathsf{PRF.Eval}(\widetilde{K}_1^{[z]}, z)$. Since this is the only difference between the two proofs, these are functionally equivalent, because the functionality of the PRF at non-punctured points is preserved under puncturing.

For $i = z + 1$, the difference between the two circuits is that $G_{z+1}^{3.1}$ uses a hardcoded proof $\pi^\star$ to calculate $\pi$, where $\pi^\star$ is calculated using a ciphertext $\mathsf{ct}^\star$ and randomness $r^\star$. We have that

$$
\begin{aligned}
G_z^{3.4}(z+1, j) &= \left(\mathsf{LDC}(\mathsf{ct}, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}, z+1, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk}), w; \mathsf{PRF.Eval}(\widetilde{K}_1^{[z]}, z+1))\right)_j \\
&= \left(\mathsf{LDC}(\mathsf{ct}^\star, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}^\star, z+1, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk}), w; \mathsf{PRF.Eval}(\widetilde{K}_1^{[z]}, z+1))\right)_j \\
&= \left(\mathsf{LDC}(\mathsf{ct}^\star, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}^\star, z+1, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk}), w; \mathsf{PRF.Eval}(K_1, z+1))\right)_j \\
&= \left(\mathsf{LDC}(\mathsf{ct}^\star, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}^\star, z+1, \mathsf{com}_{C_x^b}, \mathsf{com}_1, \mathsf{pk}), w; r^\star)\right)_j \\
&= \left(\mathsf{LDC}(\mathsf{ct}^\star, \pi^\star)\right)_j = G_{z+1}^{3.1}(z+1, j),
\end{aligned}
$$

where $\mathsf{ct} = \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^b, z+1); \mathsf{PRF.Eval}(K_0, z+1))$ and $w = (C_x^b, \mathsf{PRF.Eval}(K_0, z+1), r_0)$. Therefore, $G_{z+1}^{3.1}$ and $G_z^{3.4}$ are computationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

**Claim 4.9.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_{1^{\ell_1}}^{3.4}(\lambda))=1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^4(\lambda))=1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** We show that $G^{3.4}_{1^{\ell_1}}$ and $G^4$ are functionally equivalent. For any input $(i,j)$ where $i \neq 1^{\ell_1}$, the difference between these hybrids is that the proof generated in $G^{3.4}_{1^{\ell_1}}(i,j)$ is calculated using the punctured PRF key $K_1^{[1^{\ell_1}]}$, and the proof in $G^4(i,j)$ is calculated using the non-punctured key $K_1$. Both are equivalent because neither uses the PRF on the punctured point.

When $i = 1^{\ell_1}$, we have that the ciphertexts $\mathsf{ct}$ generated by both circuits are both $\mathsf{IFE.Enc}(\mathsf{pk}, (C^b_x, i); \mathsf{PRF.Eval}(K_0, i))$, and thus are the same. For the proof $\pi$, the circuit $G^{3.4}_{1^{\ell_1}}$ has a hardcoded proof $\pi^\star$ which is calculated using the ciphertext $\mathsf{ct}^\star$. This ciphertext $\mathsf{ct}^\star$ is identical to the ciphertext $\mathsf{ct}$ generated by $G^4$, and both proofs are for the same statement and witness. Moreover, $\pi^\star$ uses the unpunctured key $K_1$ to generate the randomness for the proof, just as $\pi$ generated by $G^4$. Therefore, both the ciphertexts and proofs generated by both circuits are identical, and thus the circuits are functionally equivalent. Therefore, these are computationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

**Claim 4.10.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}^4(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^5(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This is due to the computational hiding property of the commitment scheme. In particular, the commitment $\mathsf{com}_{C^b_x}$ is hardwired into the circuit $G^4$ and is part of the statement for the Multi-NIZK proof but the opening $r_0$ to $\mathsf{com}_{C^b_x}$ is not used, and in particular is not included in the circuit $G^4$. Therefore, if one could distinguish between these hybrids, it would break the hiding property of the commitment scheme. Therefore, these are computationally indistinguishable. □

**Claim 4.11.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}^5(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^{6.1}_{0^{\ell_1}}(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** We show that the two circuits $G^5$ and $G^{6.1}_{0^{\ell_1}}$ agree on all inputs $(i,j)$. When $i \neq 0^{\ell_1}$, the difference between the two circuits is that $G^5$ uses the unpunctured PRF key $K_0$ to generate the randomness for the ciphertext $\mathsf{ct}$ while $G^{6.1}_{0^{\ell_1}}$ uses the punctured key $\widetilde{K}_0^{[0^{\ell_1}]}$ to generate the randomness for $\mathsf{ct}$. Since neither circuit evaluates the PRF on the punctured point when $i \neq 0^{\ell_1}$, it holds that $G^5(i,j) = G^{6.1}_{0^{\ell_1}}(i,j)$ for $i \neq 0^{\ell_1}$ because functionality is preserved under puncturing.

For $i = 0^{\ell_1}$, the difference between the two circuits is that $G^{6.1}_{0^{\ell_1}}$ uses a hardcoded ciphertext $\mathsf{ct}^\star$ to compute the ciphertext $\mathsf{ct}$, which is equivalent to $\mathsf{ct}$ generated by $G^5$. In particular, we have that

$$\begin{aligned} G^5(0^{\ell_1}, j) &= \Big(\mathsf{ct}, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}, i, \mathsf{com}_{C^0_x}, \mathsf{com}_1, \mathsf{pk}), r_1; \mathsf{PRF.Eval}(K_1, 0^{\ell_1})\Big)_j \\ &= \Big(\mathsf{ct}^\star, \mathsf{NIZK.P}(\sigma, (\mathsf{ct}^\star, i, \mathsf{com}_{C^0_x}, \mathsf{com}_1, \mathsf{pk}), r_1; \mathsf{PRF.Eval}(K_1, 0^{\ell_1})\Big)_j = G^{6.1}_{0^{\ell_1}}(0^{\ell_1}, j) \end{aligned}$$

where $\mathsf{ct} = \mathsf{IFE.Enc}(\mathsf{pk}, (C^0_x, i; \mathsf{PRF.Eval}(K_0, 0^{\ell_1}))$. Therefore, $G^5$ and $G^{6.1}_{0^{\ell_1}}$ are functionally equivalent, so these hybrids are computationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

**Claim 4.12.** *For every $z \in \{0,1\}^{\ell_1}$, for any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}^{6.1}_z(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^{6.2}_z(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the pseudorandom property of the PRFs at punctured points. The difference between these two hybrids is that $\mathsf{Hyb}^{6.1}_z(\lambda)$ calculates $r^\star = \mathsf{PRF.Eval}(K_0, z)$ and $\mathsf{Hyb}^{6.2}_z(\lambda)$ calculates $r^\star$ as a truly random value. If there were an adversary $\mathcal{A}$ that could distinguish between these two hybrids, we could construct an adversary $\mathcal{B}$ that receives the punctured key $\widetilde{K}_0^{[z]}$ and a value $\hat{r}$ and constructs the circuits $G^{6.1}_z$ and $G^{6.2}_z$ uses $\hat{r}$ as $r^\star$. Then, the distinguishing advantage of $\mathcal{A}$ would translate exactly into the distinguishing advantage of $\mathcal{B}$ in breaking the security of the puncturable PRF. Therefore, these are computationally indistinguishable. □

**Claim 4.13.** *For all $z \in \{0,1\}^{\ell_1}$, for any PPT adversary $\mathcal{A}$, it holds that* $\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{6.2}(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{6.3}(\lambda)) = 1\right] \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the $\lambda$-selective security of $\mathsf{IFE}$. In particular, the difference between these two hybrids is the hardcoded ciphertext $\mathsf{ct}^\star$ in $G_z^{6.2}$ and $G_z^{6.3}$. In the first, the hardcoded ciphertext $\mathsf{ct}^\star = \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^b, i);\ r^\star)$ and in the second, $c^\star = \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^0, i);\ r^\star)$. We show that these hybrids are computationally indistinguishable in two steps. First, we show that $\mathsf{Hyb}_z^{6.2}(\lambda)$ is indistinguishable from an intermediate hybrid $\mathsf{Hyb}_z^{6.2.5}(\lambda)$ which is the same as $\mathsf{Hyb}_z^{6.2}(\lambda)$, except that we change $\mathsf{ct}^\star$ to $\mathsf{IFE.Enc}(\mathsf{pk}, 0^{s+\ell_1}; r^\star)$. We show that if there exists an adversary $\mathcal{A}$ that can distinguish between $\mathsf{Hyb}_z^{6.2}(\lambda)$ and $\mathsf{Hyb}_z^{6.2.5}(\lambda)$ with noticeable probability, there exists an adversary $\mathcal{B}$ that breaks the selective security of $\mathsf{IFE}$.

For any set of $\lambda$ circuits $\{F_k\}_{k\in[\lambda]}$ (in the circuit class for $\mathsf{IFE}$) with $F_k(C_x^0, i) = F_k(C_x^0, i)$ for all $k$, and with $F_0 = \mathcal{U}$, let $\mathsf{sk}_k = \mathsf{IFE.Keygen}(\mathsf{msk}, F_k)$ for each $k$. The adversary $\mathcal{B}$ acts as follows. $\mathcal{B}$ receives $(\mathsf{pk}, \{F_k\}, (C_x^b, i), 0^{n+\ell_1}, \{\mathsf{sk}_k\})$, and a challenge ciphertext $\hat{\mathsf{ct}}$ from the challenger, where $\hat{\mathsf{ct}}$ is either an encryption of $C_x^b$ or of $0^{n+\ell_1}$. We also let $C_x^0$ and $C_x^1$ be given to $\mathcal{B}$. Then, $\mathcal{B}$ generates $K_0, \widetilde{K}_0^{[z]}, K_1, \sigma, r_0, r_1, \mathsf{com}_{C_x^0}$, and $\mathsf{com}_1$ as in $\mathsf{Hyb}_z^{6.2}(\lambda)$. Then, $\mathcal{B}$ generates a circuit $G'$ following the description of $G_z^{6.2}$ in $\mathsf{Hyb}_z^{6.2}(\lambda)$, with the only difference being that the hardwired ciphertext $\mathsf{ct}^\star$ is set to the challenge ciphertext $\hat{\mathsf{ct}}$. $\mathcal{B}$ then obfuscates $G'$ to obtain $\widetilde{G}$ and sends $\mathsf{pk}, \mathsf{sk}_0, (\widetilde{G}', \mathsf{com}_{C_x^0}, \mathsf{com}_1, \sigma)$ to $\mathcal{A}$ as the output of $\mathsf{FE.Enc}(\mathsf{pk}, C_x^b)$. It is easy to see that if $\hat{\mathsf{ct}}$ is an encryption of $C_x^b$, then we are in $\mathsf{Hyb}_z^{6.2}(\lambda)$, and if $\hat{c}$ is an encryption of $0^{n+\ell_1}$, then we are in $\mathsf{Hyb}_z^{6.2.5}(\lambda)$. Therefore, the distinguishing advantage of $\mathcal{B}$ is the same as that of $\mathcal{A}$, thereby breaking the selective security of $\mathsf{IFE}$. The same proof holds to show that $\mathsf{Hyb}_z^{6.2.5}(\lambda)$ and $\mathsf{Hyb}_z^{6.3}(\lambda)$ are computationally indistinguishable, except that $\hat{c}$ will either be an encryption of $0^{n+\ell_1}$ or of $C_x^0$. Therefore, this shows that $\mathsf{Hyb}_z^{6.2}(\lambda)$ and $\mathsf{Hyb}_z^{6.3}(\lambda)$ are computationally indistinguishable by the security of $\mathsf{IFE}$. □

**Claim 4.14.** *For every $z \in \{0,1\}^{\ell_1}$, for any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{6.3}(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{6.4}(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the pseudorandom property of the PRFs at punctured points. The difference between these two hybrids is that $\mathsf{Hyb}_z^{6.3}(\lambda)$ calculates $r^\star$ as a truly random value and $\mathsf{Hyb}_z^{6.4}(\lambda)$ calculates $r^\star = \mathsf{PRF.Eval}(K_0, z)$. If there were an adversary $\mathcal{A}$ that could distinguish between these two hybrids, we could construct an adversary $\mathcal{B}$ that receives the punctured key $\widetilde{K}_0^{[z]}$ and a value $\hat{r}$ and constructs the circuits $G_z^{6.3}$ and $G_z^{6.4}$ uses $\hat{r}$ as $r^\star$. Then, the distinguishing advantage of $\mathcal{A}$ would translate exactly into the distinguishing advantage of $\mathcal{B}$ in breaking the security of the puncturable PRF. Therefore, these are computationally indistinguishable. □

**Claim 4.15.** *For every $z \in \{0,1\}^{\ell_1} \setminus \{1^{\ell_1}\}$, for any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathsf{Hyb}_z^{6.4}(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}_{z+1}^{6.1}(\lambda)) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** This holds due to the security of $\mathsf{xi}\mathcal{O}$. Specifically, it is easy to see that the two circuits $G_z^{6.4}$ and $G_{z+1}^{6.1}$ agree on all inputs $(i, j)$ where $i \notin \{z, z+1\}$ because the functionality of the PRF at non-punctured points is preserved under puncturing.

When $i = z$, the difference between the two circuits is that $G_z^{6.4}(z, j)$ uses a hardcoded ciphertext $\mathsf{ct}^\star$ to generate the ciphertext. We have that

$$\begin{aligned}\mathsf{ct}^\star &= \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^0, z); \mathsf{PRF.Eval}(K_0, z)) \\ &= \mathsf{IFE.Enc}(\mathsf{pk}, (C_x^b, z+1); \mathsf{PRF.Eval}(\widetilde{K}_0^{[z+1]}, z)) = \mathsf{ct}\end{aligned}$$

where $\mathsf{ct}$ is the ciphertext generated by $G^{6.1}_{z+1}(z,j)$. This implies that the proofs $\pi$ generated by both circuits are the same, because the only difference between the proofs is the use of $\mathsf{ct}$ in the statement being proven.

For $i = z+1$, we have a similar argument. The difference between the two circuits is that $G^{6.1}_{z+1}(z+1,j)$ uses a hardcoded ciphertext $\mathsf{ct}^\star$ to generate the ciphertext. We have that

$$\begin{aligned}\mathsf{ct}^\star &= \mathsf{IFE.Enc}(\mathsf{pk}, (C^b_x, z+1); \mathsf{PRF.Eval}(K_0, z+1))\\ &= \mathsf{IFE.Enc}(\mathsf{pk}, (C^b_x, z+1); \mathsf{PRF.Eval}(\widetilde{K}_0^{[z]}, z+1)) = \mathsf{ct}\end{aligned}$$

where $\mathsf{ct}$ is the ciphertext generated by $G^{6.4}_z(z+1,j)$. As above, this implies that the proofs $\pi$ generated by both circuits are identical. Therefore, these circuits are functionally equivalent, so the hybrids are computationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

**Claim 4.16.** *For any PPT $\mathcal{A}$, it holds that* $\Pr\left[\mathcal{A}(\mathsf{Hyb}^{6.4}_{1^{\ell_1}}(\lambda)) = 1\right] - \Pr\left[\mathcal{A}(\mathsf{Hyb}^{7}_{0}(\lambda)) = 1\right] \leq \mathsf{negl}(\lambda)$ *for a negligible function* $\mathsf{negl}$.

**Proof.** We show that the two circuits $G^{6.4}_{1^{\ell_1}}$ and $G^7$ agree on all inputs $(i,j)$.

When $i \neq 1^{\ell_1}$, the difference between the two circuits is that $G^7(i,j)$ uses the unpunctured PRF key $K_0$ to generate the randomness for the ciphertext $\mathsf{ct}$ while $G^{6.4}_{1^{\ell_1}}$ uses the punctured key $\widetilde{K}_0^{[1^{\ell_1}]}$ to generate the randomness for $\mathsf{ct}$. Since the value of the PRF on the punctured point is not needed for either of these computations, it holds that $G^{6.4}_{1^{\ell_1}}(i,j) = G^7(i,j)$ because functionality is preserved under puncturing.

For $i = 1^{\ell_1}$, the difference between the two circuits is that $G^{6.4}_z(1^{\ell_1}, j)$ calculates the ciphertext as a hardwired ciphertext $\mathsf{ct}^\star$. We have that

$$\mathsf{ct}^\star = \mathsf{IFE.Enc}(\mathsf{pk}, (C^0_x, 1^{\ell_1}); \mathsf{PRF.Eval}(K_0, 1^{\ell_1})) = \mathsf{ct}$$

where $\mathsf{ct}$ is the ciphertext generated by $G^7$. Therefore, $G^{6.4}_{1^{\ell_1}}$ and $G^7$ are functionally equivalent, so these hybrids are compuatationally indistinguishable by the security of $\mathsf{xi}\mathcal{O}$. □

By considering the sequence of hybrids, we conclude that probability of distinguishing between $\mathsf{Hyb}^1(\lambda)$ and $\mathsf{Hyb}^7(\lambda)$ is $(6+9\cdot 2^{\ell_1})\cdot\mathsf{negl}(\lambda) \leq \mathsf{poly}(\lambda)\cdot\mathsf{negl}(\lambda)$ which is negligible in $\lambda$ for a negligible function $\mathsf{negl}$. Since $\mathsf{Hyb}^1(\lambda)$ is the real experiment, the probability of outputting $b$ in the real experiment is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$. Recall that this shows that the probability of any PPT adversary outputting $b$ on input $(C^0, C^1, \mathsf{pk}, \mathsf{sk}_{\mathcal{U}}, \widetilde{C}^x)$ is at most $\frac{1}{2} + \mathsf{negl}(\lambda)$.

We now conclude the proof by showing that no PPT adversary can output $b$ on input $(C^0, C^1, \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b))$ with probability negligibly far from $\frac{1}{2}$. Define a sequence of hybrids $\mathcal{H}^x(\lambda)$ for $x \in \{0,1\}^{n-\frac{n}{d}}$, as follows:

- $\boldsymbol{\mathcal{H}^{0^{n-\frac{n}{d}}}(\lambda)}$ : This is the real experiment, where $b \leftarrow \{0,1\}$ and the adversary receives $(C^0, C^1, \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b))$. In particular, in the calculation of $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C^b))$, for each $x$, $\widetilde{C}^x$ is generated as in $\mathsf{Hyb}^0(\lambda)$, i.e., as an obfuscation corresponding to $C^b_x$.
- $\boldsymbol{\mathcal{H}^x(\lambda)}$ **for** $\boldsymbol{x \in \{0,1\}^{n-\frac{n}{d}} \setminus \left\{0^{n-\frac{n}{d}}\right\}}$ : This hybrid is obtained from $\mathcal{H}^{x-1}(\lambda)$ by replacing $\widetilde{C}^x$ with the value of $\widetilde{C}^x$ generated according to $\mathsf{Hyb}^7(\lambda)$, i.e., $\widetilde{C}^x$ is now independent of $b$.

Observe that $\mathcal{H}^{0^{n-\frac{n}{d}}}(\lambda)$ corresponds to the real experiment and $\mathcal{H}^{1^{n-\frac{n}{d}}}(\lambda)$ is independent of $b$. We now show the following claim.

**Claim 4.17.** *For any PPT $\mathcal{A}$, it holds that* $\left|\Pr\left[\mathcal{A}(\mathcal{H}^x(\lambda) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{H}^{x+1}) = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for every* $x \in \{0,1\}^{n-\frac{n}{d}} \setminus \left\{1^{n-\frac{n}{d}}\right\}$ *for a negligible function* $\mathsf{negl}$.

**Proof.** Let $x \in \{0,1\}^{n-\frac{n}{d}} \setminus \left\{1^{n-\frac{n}{d}}\right\}$ and suppose for contradiction that there exists an adversary $\mathcal{A}$ and polynomial $p$ such that for infinitely many values of $\lambda$, $\mathcal{A}$ can distinguish between $\mathcal{H}^x(\lambda)$ and $\mathcal{H}^{x+1}(\lambda)$ with probability $\frac{1}{p(\lambda)}$. We construct an adversary $\mathcal{B}$ that can distinguish between $\mathsf{Hyb}^0(\lambda)$ and $\mathsf{Hyb}^7(\lambda)$ corresponding to $x+1$.

$\mathcal{B}$ receives as input $(\mathsf{pk}, \mathsf{sk}_{\mathcal{U}}, \widetilde{C}^\star)$ where $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{IFE.Setup}(1^\lambda)$ and $\mathsf{sk}_{\mathcal{U}} \leftarrow \mathsf{IFE.Keygen}(\mathsf{msk}, \mathcal{U})$, as in both $\mathsf{Hyb}^0(\lambda)$ and $\mathsf{Hyb}^7(\lambda)$. The value $\widetilde{C}^\star$ either corresponds $C^b_{x+1}$ according to $\mathsf{Hyb}^0(\lambda)$ or $C^0_{x+1}$ according to $\mathsf{Hyb}^7(\lambda)$. Then, $\mathcal{B}$ chooses a bit $b' \leftarrow \{0,1\}$. Then, for all $x' < x+1$, $\mathcal{B}$ uses $\mathsf{pk}$ to generate $\widetilde{C}^{x'}$ according to $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x']$ as in $\mathsf{Hyb}^7(\lambda)$, and for all $x' > x+1$, $\mathcal{B}$ uses $\mathsf{pk}$ to generate $\widetilde{C}^{x'}$ according to $\mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)[\mathsf{pk}, x']$ as in $\mathsf{Hyb}^1(\lambda)$ using $C^{b'}_{x'}$. Then, $\mathcal{B}$ sets $\widetilde{C}^{x+1} = \widetilde{C}^\star$ and sends $\left(\mathsf{pk}, \mathsf{sk}_{\mathcal{U}}, \left\{\widetilde{C}^x\right\}_{x \in \{0,1\}^{n-\frac{n}{d}}}\right)$ to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs the response $b''$ that $\mathcal{B}$ receives from $\mathcal{A}$.

Observe that if $b' = b$, then if $\widetilde{C}^\star$ corresponds to $\mathsf{Hyb}^0(\lambda)$ then $\mathcal{A}$'s input is distributed exactly as $\mathcal{H}^x(\lambda)$, and if $\widetilde{C}^\star$ corresponds to $\mathsf{Hyb}^7(\lambda)$, then $\mathcal{A}$'s input is distributed according to $\mathcal{H}^{x+1}(\lambda)$. Therefore, if $b' = b$, then $\mathcal{B}$ succeeds with probability $\frac{1}{p(\lambda)}$. Therefore, $\mathcal{B}$ has advantage $\frac{1}{2p(\lambda)}$ in distinguishing between $\mathsf{Hyb}^0(\lambda)$ and $\mathsf{Hyb}^7(\lambda)$ corresponding to $x+1$, which is a contradiction. Therefore, $\mathcal{H}^x(\lambda)$ is computationally indistinguishable from $\mathcal{H}^{x+1}(\lambda)$. □

Therefore, no adversary can distinguish $\mathcal{H}^{0^{n-\frac{n}{d}}}$ and $\mathcal{H}^{1^{n-\frac{n}{d}}}$ with probability more than $2^{n-\frac{n}{d}} \cdot \mathsf{negl}(\lambda)$ for a negligible function $\mathsf{negl}$. Since $n \in O(\log(\lambda))$, this is negligible in $\lambda$, thus concluding the proof of security for $\mathsf{xi}\mathcal{O}'$.

□

### 4.2 $(1/\mathsf{poly} - \mathsf{negl})$-Worst Case XiO to $(1 - \mathsf{negl})$-Worst Case XiO

In this section, we show how to modify the construction of our $(1/\mathsf{poly}(\lambda) - \mathsf{negl}(\lambda))$-worst-case correct XiO to obtain a $(1 - \mathsf{negl}(\lambda))$-worst-case correct XiO. This transformation involves creating many parallel repetitions of the given XiO scheme, such that one of them will be correct with high probability. This correctness of the resulting scheme relies on the fact that we can *identify* repetitions that did not succeed. Let $\mathsf{xi}\mathcal{O}$ be the $\left(\frac{\gamma}{16} - \mathsf{negl}(\lambda)\right)$-worst-case correct XiO scheme resulting from the above transformation, for any class of circuits $\mathcal{C}^{s,n} \in \mathsf{P}^{\log}$. We define the almost perfectly correct scheme $\mathsf{xi}\mathcal{O}'$ as follows. This scheme is parametrized by $N = \frac{16\lambda}{\gamma}$.

**$(1/\mathsf{poly})$-worst-case correct XiO to $(1 - \mathsf{negl})$-worst-case correct XiO**:

- $\widetilde{C} \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ :
  1. For each $z \in [N]$, let $\widetilde{C}^z \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, C)$
  2. Output $\left\{\widetilde{C}^z\right\}_{z \in [N]}$.
- $y \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Eval}(\widetilde{C}, x)$ :
  1. For every $z \in [N]$, run $y^z = \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{C}^z, x)$. Let $z^\star$ be the first index for which $y^z \neq \bot$.
  2. Output $y^{z^\star}$, or $\bot$ if $y^{z^\star}$ is not defined.

**Claim 4.18.** *Let $p(\cdot)$ be any polynomial. If there exists a $(\frac{1}{16p(\lambda)} - \mathsf{negl}(\lambda))$-worst-case correct XiO scheme for a circuit class $\mathcal{C}^{s,n}$ for some negligible function $\mathsf{negl}$, then there exists a $(1 - \mathsf{negl}'(\lambda))$-worst-case correct XiO scheme for $\mathcal{C}^{s,n}$ for some negligible function $\mathsf{negl}'$.*

**Proof.** Let $\mathsf{xi}\mathcal{O}$ be the $\left(\frac{\gamma}{16} - \mathsf{negl}(\lambda)\right)$-worst-case correct XiO scheme from the transformation in Section 4.1 for $\mathcal{C}^{s,n} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$, where $\gamma = \frac{1}{p(\lambda)}$. Let $\mathsf{xi}\mathcal{O}'$ be the resulting scheme in the above

construction. The efficiency and security of $\mathsf{xi}\mathcal{O}'$ follow directly from the fact that $\mathsf{xi}\mathcal{O}'$ consists of polynomially many parallel repetitions of $\mathsf{xi}\mathcal{O}$. Therefore, we focus on showing $(1 - \mathsf{negl}(\lambda))$-worst-case correctness.

**Worst-Case Correctness.** To show correctness of $\mathsf{xi}\mathcal{O}'$, consider the probability that for $C \in \mathcal{C}_\lambda$ and $x \in \{0,1\}^n$, $\widetilde{C} \leftarrow \mathsf{xi}\mathcal{O}'.\mathsf{Obf}(1^\lambda, C)$ and evaluation of $\widetilde{C}$ on $x$ succeeds. The evaluation algorithm $\mathsf{xi}\mathcal{O}'.\mathsf{Eval}(\widetilde{C}, x)$ starts by running $y^z \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{C}^z, x)$ for each $z \in [N]$, and selects the first index $z^\star$ for which $y^{z^\star} \neq \bot$; we set $z^\star = \bot$ if no such index exists. We want to show that with high probability, such an index $z^\star$ exists and $y^{z^\star} = C(x)$. Let $r_z$ denote the randomness used by the $z$th obfuscation.

We first show that the probability that $z^\star = \bot$ is small. Let $X_z = 1$ if $y^z = \bot$, and let $X_z = 0$ otherwise. Then,

$$\Pr_{r_z}[X_z = 1] = \Pr_{r_z}[y^z = \bot] \leq \Pr_{r_z}[y^z \neq C(x)] \leq 1 - \frac{\gamma}{16} - \mathsf{negl}(\lambda)$$

by the worst-case correctness of $\mathsf{xi}\mathcal{O}$. Therefore, since the repetitions are independent,

$$\begin{aligned}\Pr[z^\star = \bot] = \Pr[\forall z, X_z = 1] &= (\Pr[X_1 = 1])^{\frac{16\lambda}{\gamma}} \\ &\leq \left(1 - \frac{\gamma}{16} - \mathsf{negl}(\lambda)\right)^{\frac{16\lambda}{\gamma}} \leq \left(1 - \frac{\gamma}{16}\right)^{\frac{16\lambda}{\gamma}} \leq \frac{1}{e^\lambda},\end{aligned}$$

so the probability that $z^\star = \bot$ is negligible in $\lambda$.

We now show that for any $z$ for which $y^z \neq \bot$, it holds that $y^z = C(x)$ with high probability. To do so, we briefly recall the $\mathsf{xi}\mathcal{O}$ evaluation algorithm and introduce notation for the $z$th instance. The algorithm $\mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{C}^z, x)$ does the following:

1. Parse $x = x_1 x_2$ with $|x_1| = n - \frac{n}{d}$ and evaluate the obfuscated circuit $\widetilde{G}^{x_1,z}$ on all inputs $(i,j)$ to obtain $c_{ij}^z = \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{G}^{x_1,z}, (i,j))$.
2. For each $i$, run $\mathsf{LDC.Dec}(c_{i1}^z \cdots c_{i2^{\ell_2}}^z)$ to obtain a list of $(c_i^{k,z}, \pi_i^{k,z})$ for polynomially many $k$.
3. For each $i$ and each $k$, run $\mathsf{NIZK.V}(\sigma, x_i^{k,z}, \pi_i^{k,z})$ to check if the statement $x_i^{k,z} \in L$, where $x_i^{k,z} = (\mathsf{ct}_i^{k,z}, i, \mathsf{com}_{C_{x_1}}^{x_1,z}, \mathsf{com}_0^{x_1,z}, \mathsf{pk}_z)$. For the first index $k$ for which the verification passes, set $\mathsf{ct}_i^z = \mathsf{ct}_i^{k,z}$.
4. For each $i$, decrypt to obtain $y_i^z = \mathsf{IFE.Dec}(\mathsf{sk}_{\mathcal{U}_z}, \mathsf{ct}_i^z)$.
5. Let $y^z$ be the $x_2$th element of $\mathsf{ECC.Dec}(y_1^z \cdots y_{2^{\ell_1}}^z)$ and output $y^z$.

Therefore, for any $z$, we have that

$$\begin{aligned}\Pr[y^z \neq C(x) \mid y^z \neq \bot] &\leq \Pr\left[\mathsf{ECC.Dec}(y_1^z \cdots y_{2^{\ell_1}}^z) \neq T(C_{x_1}) \mid y^z \neq \bot\right] \\ &= \Pr[\exists i : y_i^z \neq \bot \ \wedge \ y_i^z \neq \mathsf{ECC}(T(C_{x_1}))[i] \mid y^z \neq \bot],\end{aligned}$$

where the last equality holds because $y^z \neq \bot$ implies that there are sufficiently many $y_i^z$ which are not $\bot$, and thus $\mathsf{ECC.Dec}$ could only output the wrong answer due to an index $i$ that is incorrect. Then, by the correctness of $\mathsf{IFE}$, we have that for some negligible function $\mathsf{negl}$, this is

$$\begin{aligned}&\Pr[\exists i : y_i^z \neq \bot \ \wedge \ y_i^z \neq \mathsf{ECC}(T(C_{x_1}))[i] \mid y^z \neq \bot] \\ &\leq \Pr[\exists i : \mathsf{ct}_i^z \neq \bot \ \wedge \ \forall r : \mathsf{ct}_i^z \neq \mathsf{IFE.Enc}(\mathsf{pk}_z, (C_{x_1}, i); r) \mid y^z \neq \bot] + \mathsf{negl}(\lambda) \\ &= \Pr\left[\exists i : \mathsf{NIZK.V}(\sigma, x_i^{k,z}, \pi_i^{k,z}) = 1 \ \wedge \ \forall r : \mathsf{ct}_i^z \neq \mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i); r) \mid y^z \neq \bot\right] + \mathsf{negl}(\lambda) \\ &\leq \mathsf{negl}(\lambda)\end{aligned}$$

by the soundness of $\mathsf{NIZK}$, because if $c_i^z \neq \mathsf{IFE.Enc}(\mathsf{pk}, (C_{x_1}, i); r)$ for all $r$ then $x_i^z \notin L$. Therefore, we conclude that

$$\begin{aligned}\Pr\left[y^{z^\star} \neq C(x)\right] &\leq \Pr\left[y^{z^\star} \neq C(x) \wedge y^{z^\star} \neq \bot\right] + \Pr\left[y^{z^\star} = \bot\right] \\ &\leq \Pr\left[\exists z : y^z \neq C(x) \mid y^z \neq \bot\right] + \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda)\end{aligned}$$

because there are polynomially many indices $z$. Therefore, $\Pr\left[y^{z^\star} = C(x)\right] \geq 1 - \mathsf{negl}(\lambda)$, thereby showing $(1 - \mathsf{negl}(\lambda))$-worst-case correctness of $\mathsf{xi}\mathcal{O}'$.

□

### 4.3 $(1 - \mathsf{negl})$-Worst Case Correct XiO to Perfectly Correct XiO

**Claim 4.19.** *Let $\mathsf{xi}\mathcal{O}$ be a $(1-\mathsf{negl}(\lambda))$-worst case correct XiO scheme for the class of circuits $\mathcal{C}^{s,n}$. Then, there exists a perfectly correct XiO scheme for the class of circuits $C^{s,n}$.*

**Proof.** Let $\mathsf{xi}\mathcal{O}$ be the scheme for $\mathcal{C}^{s,n} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$. We show that $\mathsf{xi}\mathcal{O}$ can be amplified to a scheme $\mathsf{xi}\mathcal{O}'$ which satisfies almost perfect correctness, security, and compression, and that $\mathsf{xi}\mathcal{O}'$ can then be amplified to a scheme $\mathsf{xi}\mathcal{O}^\star$ which is perfectly correct. Let $s = s(\lambda)$ and $n = n(\lambda)$.

Given $\mathsf{xi}\mathcal{O}$, we apply a standard BPP-stype amplification. We define a new $\mathsf{xi}\mathcal{O}'$ that on input circuit $C$ runs $\mathsf{xi}\mathcal{O}$ poly-many times (in $n$), say $O(n^2)$ times, and outputs all of the obfuscations. Evaluation is done by running all of the obfuscations and then outputting the majority value. This transformation reduces the probability of being wrong on each $x \in \{0,1\}^n$ to $\mathsf{negl}(\lambda) \cdot 2^{-n}$. Now, we apply a union bound and get that

$$\Pr\left[\widetilde{C} \leftarrow \mathsf{xi}\mathcal{O}.\mathsf{Obf}(1^\lambda, C) : \forall x \in \{0,1\}^n : \mathsf{xi}\mathcal{O}.\mathsf{Eval}(\widetilde{C}, x) = C(x)\right] \geq 1 - \mathsf{negl}(\lambda),$$

thereby showing that $\mathsf{xi}\mathcal{O}'$ is almost perfectly correct. $\mathsf{xi}\mathcal{O}'$ satisfies compression and security because it consists of $\mathsf{poly}(n)$ parallel repetitions of $\mathsf{xi}\mathcal{O}$.

Now, we change the resulting $\mathsf{xi}\mathcal{O}'$ to a perfectly correct $\mathsf{xi}\mathcal{O}^\star$ as follows. After obfuscating a circuit it goes over all inputs (in time $2^n$ which is polynomial) and checks whether the obfuscation is perfectly correct. If so, it outputs this obfuscation. If not, it outputs the circuit in the clear (padded to the correct length, and modifying $\mathsf{xi}\mathcal{O}^\star.\mathsf{Eval}$ as necessary). The resulting obfuscation is clearly perfectly correct. The security of it suffers from an extra negligible factor due to the cases where the obfuscation was not correct. Overall, the final obfuscation is secure and perfectly correct.[5]

□

### 4.4 Wrapping Up – Proof of Theorem 4.1

Let $\mathcal{C}^{s,n}$ be any class in $\mathsf{P}^{\log}$. Let $\mathsf{xi}\mathcal{O}$ be a $(1/2 + 1/p(\lambda))$-approximately correct XiO scheme for all $\mathsf{P}^{\log}$. Thus, the circuits $G^x$ in the transformation in Section 4.1 can be obfuscated when the XiO resulting from that transformation is for the class $\mathcal{C}^{s,n} \in \mathsf{P}$. Then, by Theorem 4.2, Corollary 3.23, and Theorem 3.17, assuming LWE and the existence of NIZKs, there exists a $\left(\frac{1}{16p(\lambda)} - \mathsf{negl}(\lambda)\right)$-worst-case correct XiO scheme for the class $\mathcal{C}^{s,n} \in \mathsf{P}$. Then, by Claim 4.18, there exists a $(1-\mathsf{negl}(\lambda))$-worst-case correct XiO scheme for $C^{s,n}$. Then, by Claim 4.19, there exists a perfectly correct XiO scheme for $\mathcal{C}^{s,n}$. Since this holds for any class $\mathcal{C}^{s,n} \in \mathsf{P}^{\log}$, we therefore obtain perfectly correct XiO for all of $\mathsf{P}^{\log}$.

[5] While the whole proof can be applied to XiO, this last step does not work for SXiO since we cannot go over all inputs and check the correctness of the obfuscation.

# 5 Impossibility of Key Agreement from XIO and OWFs

In this section, we show a separation from XiO and one-way functions to key agreement. In particular, we present an oracle $\Gamma$ relative to which there exists a one-way function and XiO for oracle-aided circuits, but there does not exist an oracle-aided bit-agreement protocol. This separation is in largely based on [10, 11], and in particular follows the framework of black-box separations presented in [65]. We extend the model of [10, 11] to capture obfuscation for oracle-aided circuits with all possible gates, as in [48, 49]. We begin with some preliminaries.

## 5.1 Preliminaries

Throughout this section, for ease of notation we denote both the security parameter and the size of circuits by $s$. While these could be decoupled, it is natural to combine them in this way. Therefore, in this section, we use the notation $\{\mathcal{C}_{s,n}\}_{s,n\in\mathbb{N}}$ to denote the circuit class where each $C \in \mathcal{C}_{s,n}$ has size $s$ and input length $n$.

**Definition 5.1** (Oracle-Aided Circuits)**.** *We say that $\mathcal{C}$ is a class of oracle-aided circuits if it consists of circuits, represented as a directed acyclic graph, with gates that are either Boolean operations or oracle gates. Without loss of generality, we consider oracle-aided circuits that output a single bit.*

In the above definition, we note that oracle gates could output $\perp$ (not only 0 or 1). Since we intend to capture oracle-aided circuits for all functionalities, we define $\perp \vee 1$ to output 1 and we let all other boolean operations involving $\perp$ output $\perp$.[6]

**Definition 5.2** (Query types)**.** *Let $M$ be an oracle-aided algorithm with oracle access to $\Gamma$. Then, any query $Q$ that $M$ makes to $\Gamma$ is called a* direct query *of $M$. Moreover, if $\Gamma(Q)$ issues a query $Q'$ to $\Gamma$ for any such $Q$, we say that $Q'$ is an* indirect query *of $M$ caused by $Q$.*

**Definition 5.3** ($q$-Query Algorithm)**.** *We say that an algorithm $M$ with oracle access to $\Gamma$ is a $q$-query algorithm if for every $s \in \mathbb{N}$, the total number of direct queries that $M(1^s)$ makes to $\Gamma$ is at most $q(s)$, and each query made by $M(1^s)$ has size at most $q(s)$.*

We note that the above definition is without loss of generality. In particular, with regards to the size of the queries made by $M$, in this paper we consider unbounded adversaries that make a sub-exponential number of queries. However, an adversary that makes very large queries could use them to learn new oracle query-answer pairs indirectly. Therefore, the above definition captures the notion that the adversary can only learn a sub-exponential number of query-answer pairs, but may do any amount of computation on that information.

### 5.1.1 Oracle-Aided Bit Agreement

We next define oracle-aided bit agreement protocols. We are interested in protocols where both parties $\mathcal{A}$ and $\mathcal{B}$ run in polynomial time. Therefore we start by defining a PPT oracle-aided algorithm, and then continue with the definition of an oracle-aided bit agreement protocol.

**Definition 5.4** (PPT Oracle-Aided Algorithm)**.** *We say that an oracle-aided algorithm $M$ is a PPT oracle-aided algorithm with respect to an oracle $\Gamma$ if there exists polynomials $q_1, q_2, q_3$ such that for any $s \in \mathbb{N}$, it holds that $M(1^s)$ is a $q_1$-query algorithm, all queries that $M(1^s)$ makes to $\Gamma$ have query and answer size bounded by $q_2(s)$, and $M(1^s)$ runs in time $q_3(s)$.*

**Definition 5.5.** *An* oracle-aided bit agreement protocol $\Pi = (\mathcal{A}, \mathcal{B})$ *is a tuple of PPT oracle-aided algorithms relative to an oracle $\Gamma$ with the following syntax:*

[6] This formalization allows us to capture functionalities like $\mathsf{mux}$, even if an oracle gate returns $\perp$.

- $(k_{\mathcal{A}}, k_{\mathcal{B}}, T) \leftarrow \langle \mathcal{A}^{\Gamma}(1^s; r_{\mathcal{A}}), \mathcal{B}^{\Gamma}(1^s; r_{\mathcal{B}}) \rangle$**:** *For random tapes $r_{\mathcal{A}}$ and $r_{\mathcal{B}}$ (that are poly in $s$), we denote the execution of the protocol by $\langle \mathcal{A}^{\Gamma}(1^s; r_{\mathcal{A}}), \mathcal{B}^{\Gamma}(1^s; r_{\mathcal{B}}) \rangle$. In the output, $k_{\mathcal{A}}$ is the output bit of $\mathcal{A}$, $k_{\mathcal{B}}$ is the output bit of $\mathcal{B}$, and $T$ is the protocol transcript, consisting of messages exchanged between $\mathcal{A}$ and $\mathcal{B}$.*

*We require that the following conditions hold:*

- **Perfect Completeness.** *For any $s \in \mathbb{N}$, it holds that*

$$\Pr_{r_{\mathcal{A}}, r_{\mathcal{B}}} \left[ (k_{\mathcal{A}}, k_{\mathcal{B}}, T) \leftarrow \langle \mathcal{A}^{\Gamma}(1^s; r_{\mathcal{A}}), \mathcal{B}^{\Gamma}(1^s; r_{\mathcal{B}}) \rangle : k_{\mathcal{A}} = k_{\mathcal{B}} \right] = 1.$$

- **Security.** *For any PPT oracle-aided algorithm $E$, there exists a negligible function* $\mathsf{negl}$ *such that for all sufficiently large $s \in \mathbb{N}$,*

$$\mathsf{Adv}^{\mathsf{KA}}_{\Gamma,\Pi,E}(s) \stackrel{\mathsf{def}}{=} \left| \Pr \left[ \mathsf{Exp}^{\mathsf{KA}}_{\Gamma,\Pi,E}(s) = 1 \right] - \frac{1}{2} \right| \leq \mathsf{negl}(s),$$

*where the experiment $\mathsf{Exp}^{\mathsf{KA}}_{\Gamma,\Pi,E}(s)$ is defined as follows:*

1. $(k_{\mathcal{A}}, k_{\mathcal{B}}, T) \leftarrow \langle \mathcal{A}^{\Gamma}(1^s), \mathcal{B}^{\Gamma}(1^s) \rangle$.
2. $k' \leftarrow E^{\Gamma}(1^s, T)$.
3. *If $k' = k_{\mathcal{A}}$ then output 1, otherwise output 0.*

### 5.1.2 XiO for Oracle-Aided Circuits

We now define XiO relative to an oracle, similar to the definition of iO relative to an oracle given in [10, 11]. We strengthen the [10, 11] framework to capture XiO for circuits which may contain all possible oracle gates. We first need the definition of functional equivalence relative to an oracle.

**Definition 5.6.** *Let $C_0$ and $C_1$ be two oracle-aided circuits relative to an oracle $\Gamma$. We say that $C_0$ and $C_1$ are* functionally equivalent relative to $\Gamma$*, denoted $C_0^{\Gamma} \equiv C_1^{\Gamma}$, if for all inputs $x$ it holds that $C_0^{\Gamma}(x) = C_1^{\Gamma}(x)$.*

**Definition 5.7.** *A perfectly correct XiO scheme relative to an oracle $\Gamma$ for a class $\mathcal{C} = \{\mathcal{C}_{s,n}\}_{s,n \in \mathbb{N}}$ of oracle-aided circuits is a tuple of oracle-aided algorithms $\mathsf{xi}\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ with the following syntax:*

- $\widetilde{C} \leftarrow \mathsf{Obf}^{\Gamma}(1^s, C)$**:** *The obfuscator receives the security parameter $1^s$ and a circuit $C \in \mathcal{C}_s$ and outputs a circuit $\widetilde{C}$.*
- $\mathsf{Eval}^{\Gamma}(\widetilde{C}, x)$**:** *The evaluator receives a circuit $\widetilde{C}$ and an input $x$, and outputs a string $y$ or $\perp$.*

*We require the following conditions to hold:*

- **Perfect Correctness.** *For all $s, n \in \mathbb{N}$ and all $C \in \mathcal{C}_{s,n}$ it holds that*

$$\Pr \left[ \widehat{C} \leftarrow \mathsf{Obf}^{\Gamma}(1^s, C) : C^{\Gamma} \equiv \widehat{C}^{\Gamma} \right] = 1$$

- **Indistinguishability.** *For any PPT distinguisher $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every $s \in \mathbb{N}$,*

$$\mathsf{Adv}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) \stackrel{\mathsf{def}}{=} \left| \Pr \left[ \mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) = 1 \right] - \frac{1}{2} \right| \leq \mathsf{negl}(s)$$

*where the random variable $\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s)$ is defined as follows:*

- $b \leftarrow \{0,1\}$.
- $(C_0, C_1, \mathsf{state}) \leftarrow \mathcal{D}_1^\Gamma(1^s)$ *where* $|C_0| = |C_1| = s$ *and* $C_0^\Gamma \equiv C_1^\Gamma$.
- $\widehat{C} \leftarrow \mathsf{Obf}^\Gamma(1^s, C_b)$.
- $b' \leftarrow \mathcal{D}_2^\Gamma(\mathsf{state}, \widehat{C})$.
- *If* $b' = b$ *then output 1. Otherwise, output* 0.

- **Efficiency.** $\mathsf{Obf}^\Gamma$ *satisfies the required compression for XiO.*

**Exponential Security.** We say that an obfuscator relative to an oracle $\Gamma$ is *exponentially secure* if for any $q$-query adversary $\mathcal{A}$, if there exists some $0 \leq \gamma < 1$ such that $q(s) \leq 2^{\gamma s}$, then $\mathsf{Adv}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{A},\mathcal{C}}(s)$ is at most $1/q(s)$.

### 5.1.3 The Class of Reductions

In this section, we present the class of reductions that we capture. At a high level, we say that a black-box construction of oracle-aided key agreement relative to $\Gamma$ from a one-way function and XiO is a key agreement protocol with the property that any adversary that can break the security of the key agreement protocol can be used either to invert the one-way function or break the security of XiO. Moreover, this definition captures constructions from XiO for circuits which are allowed to contain all possible oracle gates.

**Definition 5.8.** *An* $(\mathcal{A}, \mathcal{B}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$*-fully black-box construction of an oracle-aided bit-agreement protocol from a one-way function* $f$ *and an XiO scheme* $\mathsf{xi}\mathcal{O}$ *for the class of oracle-aided circuits* $\mathcal{C}$ *(which may contain circuits with both* $f$ *gates and* $\mathsf{xi}\mathcal{O}$ *gates) consists of a tuple of PPT oracle-aided algorithms* $(\mathcal{A}, \mathcal{B})$*, an oracle-aided algorithm* $M$ *that runs in time* $T_M(\cdot)$*, and functions* $\epsilon_{M,1}(\cdot)$ *and* $\epsilon_{M,2}(\cdot)$ *such that the following holds:*

- **Perfect completeness:** *For any* $s \in \mathbb{N}$*, it holds that*

$$\Pr_{r_\mathcal{A}, r_\mathcal{B}}\left[(k_\mathcal{A}, k_\mathcal{B}, T) \leftarrow \langle \mathcal{A}^{f,\mathsf{xi}\mathcal{O}}(1^s; r_\mathcal{A}), \mathcal{B}^{f,\mathsf{xi}\mathcal{O}}(1^s; r_\mathcal{B})\rangle : k_\mathcal{A} = k_\mathcal{B}\right] = 1.$$

- **Black box proof of security:** *For any function* $f = \{f_s\}_{s\in\mathbb{N}}$*, any scheme* $\mathsf{xi}\mathcal{O} = (\mathsf{Obf}, \mathsf{Eval})$ *satisfying the syntax of perfectly correct XiO for the circuit class* $\mathcal{C}$*, any oracle-aided algorithm* $E$ *that runs in time* $T_E(\cdot)$*, and any function* $\epsilon_E(\cdot)$*, if*

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{KA}}_{(f,\mathsf{xi}\mathcal{O}),(\mathcal{A},\mathcal{B}),E}(s)\right] - \frac{1}{2}\right| \geq \epsilon_E(s)$$

*for infinitely many values of* $s \in \mathbb{N}$*, then either*

$$\Pr_{x\leftarrow\{0,1\}^s}\left[f_s\left(M^{f,\mathsf{xi}\mathcal{O},E}(f_s(x))\right) = f_s(x)\right] \geq \epsilon_{M,1}\left(T_E(s)\cdot\epsilon_E^{-1}(s)\right)\cdot\epsilon_{M,2}(s)$$

*for infinitely many values of* $s \in \mathbb{N}$*, or*

$$\left|\Pr\left[\mathsf{Exp}^{XiO}_{(f,\mathsf{xi}\mathcal{O}),\mathsf{xi}\mathcal{O},M^E,\mathcal{C}}(s) = 1\right] - \frac{1}{2}\right| \geq \epsilon_{M,1}\left(T_E(s)\cdot\epsilon_E^{-1}(s)\right)\cdot\epsilon_{M,2}(s)$$

*for infinitely many values of* $s \in \mathbb{N}$*.*

**The security loss function.** Since we intend to capture constructions that may be based on super-polynomial security assumptions, we allow the algorithm $M$ to run in arbitrary time $T_M(s)$ and to have an arbitrary security loss. Moreover, we distinguish between the "adversary-dependent" security-loss $\epsilon_{M,1}(T_E(s) \cdot \epsilon_E^{-1}(s))$, and "adversary-independent" security loss $\epsilon_{M,2}(s)$. See [10, 11] for further discussion.

### 5.2 Proof Overview and the Oracle $\Gamma$

We prove the following theorem.

**Theorem 5.9.** *Let $(\mathcal{A}, \mathcal{B}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of a bit-agreement protocol from a one-way function and from XiO for a class of oracle-aided circuits $\mathcal{C}$. Then, at least one of the following holds:*

- *The reduction runs in exponential time, i.e., $T_M(s) \geq 2^{\gamma s}$ for some $\gamma > 0$.*
- *The security loss is exponential, i.e., $\epsilon_{M,1}(s^d) \cdot \epsilon_{M,2}(s) \leq 2^{-s/2}$ for some constant $d \geq 1$.*

We prove Theorem 5.9 by presenting a distribution $\mathfrak{S}_\ell$ over oracles $\Gamma$ relative to which the following properties hold: (1) there does not exist a key-agreement protocol; (2) there exists an (exponentially) secure one-way function, and (3) there exists an (exponentially) secure XiO. In this section we present the distribution over oracles for which the above occur. In Section 5.3 we prove that there exists a one-way function relative to $\Gamma$, and in Section 5.4 we show the existence of XiO relative to $\Gamma$. Finally, in Section 5.5 we show that there does not exist a key-agreement protocol relative to $\Gamma$. We start by defining the distribution of oracles that we consider.

**The oracle $\Gamma$.** Let $\ell$ be a 2-ary function with $\ell(s,n) > s$. We now define the distribution $\mathfrak{S}_\ell$ over oracles $\Gamma = (f, \mathcal{O}, \mathcal{E}) = \left(\{f_s\}_{s\in\mathbb{N}}, \{\mathcal{O}_{s,n}\}_{s,n\in\mathbb{N}}, \{\mathcal{E}_{s,n}\}_{s,n\in\mathbb{N}}\right)$. In order to define $\Gamma$, for every $s \in \mathbb{N}$, let $\Gamma_{<s}$ consist of oracles in $\Gamma$ that can be queried on inputs $s' < s$. In particular, let $\Gamma_{<s} = (f_{<s}, \mathcal{O}_{<s}, \mathcal{E}_{<s})$, where $f_{<s} = \{f_{s'}\}_{s'<s}$, $\mathcal{O}_{<s} = \left\{\mathcal{O}_{s',n}\right\}_{s'<s, n\leq s'}$, and $\mathcal{E}_{<s} = \left\{\mathcal{E}_{s',n}\right\}_{s'<s, n<s'}$. We can now define $(f, \mathcal{O}, \mathcal{E})$:

- **The function $f = \{f_s\}_{s\in\mathbb{N}}$.** For every $s \in \mathbb{N}$, the function $f_s : \{0,1\}^s \to \{0,1\}^s$ is a uniformly chosen function. We will use $f$ to implement a one-way function.
- **The function $\mathcal{O} = \{\mathcal{O}_{s,n}\}_{s,n\in\mathbb{N}}$.** For every $s \in \mathbb{N}$ and $n \leq s$, the function $\mathcal{O}_{s,n} : \{0,1\}^{2s} \to \{0,1\}^{10\ell(s,n)}$ is a uniformly chosen function. Intuitively, $\mathcal{O}_{s,n}$ will receive a description of a circuit with size $s$ and input length $n$, as well as a string of length $s$ (which represents the randomness of the obfuscator), and will increase this to a uniformly chosen string of length $10\ell(s,n)$. This will be used to implement the XiO obfuscator.
- **The function $\mathcal{E} = \{\mathcal{E}_{s,n}^{\Gamma_{<s}}\}_{s\in\mathbb{N}, n\in\mathbb{N}}$.** For every $s \in \mathbb{N}$ and $n \leq s$, we define the function $\mathcal{E}_{s,n}^{\Gamma_{<s}} : \{0,1\}^{10\ell(s,n)} \times \{0,1\}^n \to \{0,1\}^*$ as follows. On input $(\widehat{C}, x) \in \{0,1\}^{10\ell(s,n)} \times \{0,1\}^n$, the function $\mathcal{E}_{s,n}^{\Gamma_{<s}}$ finds the lexicographically first oracle-aided circuit $C$ of size $s$ and input size $n$, and a string $r \in \{0,1\}^s$ such that $\mathcal{O}_{s,n}(C, r) = \widehat{C}$, and outputs $C^{\Gamma_{<s}}(x)$. In particular, $C$ may contain queries to any oracles in $\Gamma_{<s}$, and $\mathcal{E}_{s,n}$ forwards the queries to the corresponding oracles. If no such $(C, r)$ exists, it outputs $\perp$. Looking ahead, the oracle $\mathcal{E}$ will be used to implement the XiO evaluator.

We note that the above oracles are well-defined because any circuit of size $s$ can only have an oracle gate of size smaller than $s$. In particular, if any circuit $C \in \mathcal{C}_{s,n}$ has an $\mathcal{E}_{s',n'}$ gate, then the input to the oracle gate has size $\ell(s', n') > s'$ which can be at most $s$, and thus $s' < s$. If $C$ has an

$\mathcal{O}_{s',n'}$ gate, then the input to the gate has size $2s'$ which can be at most $s$, and thus $s' < s$. If $C$ has an $f_{s'}$ gate, then there are $2s'$ input and output wires in total, and thus $s' < s$. We next upper bound the number of indirect queries in an execution of a $q$-query algorithm, relative to *any* $\Gamma$ in the support of $\mathfrak{S}_\ell$.

**Claim 5.10.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for any constant $0 < \epsilon < 1$.[7] Let $\Gamma$ be any oracle in the support of $\mathfrak{S}_\ell$, and let $M$ be an oracle-aided $q$-query algorithm relative to $\Gamma$. Then, every query made by $M$ causes at most $q(s)^4$ indirect queries, and the total number of indirect queries made by $M$ is bounded by $q(s)^5$.*

**Proof.** We want to upper bound the number of indirect queries made by $M$. Suppose that $M$ makes $q = q(s)$ direct queries. By construction of $\Gamma$, the only indirect queries caused by $M$ are due to querying $\mathcal{E}$. Observe that for any $\mathcal{E}_{s,n}$ query, the maximum number of oracle gates in the circuit evaluated by $\mathcal{E}_{s,n}$ is at most $s$. Moreover, any such oracle gate which is an $\mathcal{E}_{s',n'}$ gate must have $s' < s^{\frac{1}{2}}$ and $n' < \frac{1}{\epsilon}\log(s)$, because the size of the gate must be bounded by $s$.

Therefore, consider any $\mathcal{E}_{s,n}$ query $y$ made by $M$ for $s > 1$ (if $s = 1$, there can be no indirect queries). We can view the indirect queries caused by $y$ as a tree of queries rooted at $y$, where each node containing an $\mathcal{E}$ query has a child for every oracle gate in the circuit evaluated by $\mathcal{E}$ on this query. By the above logic, for any node at depth $i$, if it is an $\mathcal{E}$ query, it corresponds to a circuit of size $s_i < s^{\frac{1}{2^i}}$ and input length $n_i < \frac{1}{\epsilon \cdot 2^{i-1}}\log(s)$. Moreover, each $\mathcal{E}$ query at depth $i$ can cause at most $s_i$ queries at the depth $i+1$. Therefore, letting $s_0 := s$ and noting that the tree can have depth at most $\log\log(s)$, an upper bound on the total number of nodes in the tree (and thus the number of indirect queries caused by a single query made by $M$) is

$$\sum_{i=0}^{\log\log(s)} \prod_{j=0}^{i} s_j < \sum_{i=0}^{\log\log(s)} \prod_{j=0}^{i} s^{\frac{1}{2^j}} = \sum_{i=0}^{\log\log(s)} s^{\sum_{j=0}^{i}\frac{1}{2^j}} < \sum_{i=0}^{\log\log(s)} s^2 = (\log\log(s)+1)\cdot s^2 \le s^4$$

Since $M$ is a $q$-query algorithm, then $s \le q$ and thus the total number of indirect queries is bounded by $q \cdot s^4 \le q^5$. □

We now bound the probability of certain bad events related to $\Gamma$. We start by bounding the probability that the oracle $\mathcal{O}_{s,n}$ is not injective, which will be helpful in the proof of Theorems 5.18 and 5.9.

**Definition 5.11.** *Let $\mathsf{injective}^{\Gamma}_{s,n}$ be the event that $\mathcal{O}_{s,n}$ is injective when $\Gamma = (f, \mathcal{O}, \mathcal{E})$ is sampled from $\mathfrak{S}_\ell$. Let $\mathsf{injective}^{\Gamma}_{\ge s} = \bigwedge_{s' \ge s, n' \le s'} \mathsf{injective}^{\Gamma}_{s',n'}$ be the event that $\mathcal{O}_{s',n'}$ is injective for all $s' \ge s$, and let $\mathsf{injective}^{\Gamma} = \bigwedge_{s,n} \mathsf{injective}^{\Gamma}_{s,n}$ be the event that $\mathcal{O}_{s,n}$ is injective for all $s, n$.*

**Claim 5.12.** *For any $s, n \in \mathbb{N}$ with $n \le s$ and any function $\ell(s,n) > s$, it holds that $\Pr\left[\neg\mathsf{injective}^{\Gamma}_{s,n}\right] \le 2^{-6s}$ and $\Pr\left[\neg\mathsf{injective}^{\Gamma}_{\ge s}\right] \le 2^{-(5s+1)}$, where the probability is over $\Gamma \leftarrow \mathfrak{S}_\ell$.*

**Proof.** We have that for any $s, n \in \mathbb{N}$ with $n \le s$,

$$\begin{aligned}\Pr_{\Gamma\leftarrow\mathfrak{S}_\ell}\left[\neg\mathsf{injective}^{\Gamma}_{s,n}\right] &\le \Pr_{\Gamma\leftarrow\mathfrak{S}_\ell}\left[\exists\, C, r, C', r' : (C,r) \ne (C',r') \ \wedge\ \mathcal{O}_{s,n}(C,r) = \mathcal{O}_{s,n}(C',r')\right] \\ &\le \binom{2^{2s}}{2} \cdot \frac{1}{2^{10\ell(s,n)}} \le \frac{1}{2^{10\ell(s,n)-4s}} \le \frac{1}{2^{10s-4s}} = 2^{-6s}\ ,\end{aligned}$$

since $\ell(s,n) > s$. Therefore, by a union bound,

$$\Pr_{\Gamma\leftarrow\mathfrak{S}_\ell}\left[\neg\mathsf{injective}^{\Gamma}_{\ge s}\right] \le \sum_{s'=s}^{\infty}\sum_{n'=1}^{s'} \Pr_{\Gamma\leftarrow\mathfrak{S}_\ell}\left[\neg\mathsf{injective}^{\Gamma}_{s',n'}\right] \le \sum_{s'=s}^{\infty}\sum_{n'=1}^{s'} \frac{1}{2^{6s'}} = \sum_{s'=s}^{\infty}\frac{s'}{2^{6s'}} \le \sum_{s'=s}^{\infty}\frac{1}{2^{5s'}} \le \frac{2}{2^{5s}}.$$

[7] Throughout this section, we will restrict $\ell(s,n) = 2^{n\epsilon}\cdot s^2$, but we note that the proof holds when $\ell(s,n) = 2^{n\epsilon}\cdot s^c$ for any constant $c > 1$.

□

We next bound the probability of an oracle-aided algorithm "guessing" a point in the image of $\mathcal{O}_{s,n}$ without receiving it as the answer to a query. This will be helpful in the proof of Theorem 5.18.

**Definition 5.13.** *Let $\ell$ be any two-ary function and let $\Gamma \leftarrow \mathfrak{S}_\ell$. For any oracle-aided algorithm $M$, let $\mathsf{spoof}^\Gamma_{s,n}$ be the event that there exists $\widetilde{C} \in \{0,1\}^{10\ell(s,n)}$ and $x \in \{0,1\}^n$ such that both of the following occur:*

1. *$\widetilde{C}$ is not the output of any direct or indirect $\mathcal{O}_{s,n}$ query made by $M(1^s)$.*
2. *$M(1^s)$ makes either a direct or indirect query to $\mathcal{E}_{s,n}$ on input $(\widetilde{C}, x)$ and $\mathcal{E}_{s,n}(\widetilde{C}, x) \neq \bot$.*

*Let $\mathsf{spoof}^\Gamma = \bigvee_{s,n} \mathsf{spoof}^\Gamma_{s,n}$.*

We now show that conditioned on $\mathsf{injective}^\Gamma_{\geq s}$, the probability that any $q$-query algorithm $M$ causes $\mathsf{spoof}^\Gamma_{s,n}$ to occur is small. This proof follows ideas similar to [47].

**Claim 5.14.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for some constant $0 \leq \epsilon < 1$. Then, for any oracle-aided $q$-query algorithm $M$, for any $(s,n)$, it holds that*

$$\Pr_{\Gamma \leftarrow \mathfrak{S}_\ell}\left[\mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] \leq \frac{2^{2s} \cdot q(s)}{2^{10s} - 2q(s)^5}.$$

We prove this claim using the following lemma, which bounds the probability of an adversary who doesn't make queries to $\mathcal{E} \setminus \{\mathcal{E}_{<s}\}$ "guessing" a point in the image of $\mathcal{O}_{s,n}$.

**Lemma 5.15.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for a constant $0 \leq \epsilon < 1$. Let $\mathcal{A}$ be a $q'$-query adversary with oracle access to $\Gamma$ that only makes queries to $f, \mathcal{O}, \mathcal{E}_{<s}$. Then, letting $\mathsf{win}^\Gamma_{s,n}$ denote the event that $\mathcal{A}(1^s)$ outputs a value $\widetilde{C}$ in the image of $\mathcal{O}_{s,n}$ without receiving $\widetilde{C}$ as the answer to any query, it holds that*

$$\Pr_{\Gamma \leftarrow \mathfrak{S}_\ell, \mathcal{A}}\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] \leq \frac{2^{2s}}{2^{10s} - q'(s)}.$$

**Proof.** When $\Gamma \leftarrow \mathfrak{S}_\ell$, the choice of $\mathcal{O}_{s,n}$ is independent of the choice of $f$, $\mathcal{O} \setminus \{\mathcal{O}_{s,n}\}$, and $\mathcal{E}_{<s}$, because each query to these oracles cannot reveal any point in the image of $\mathcal{O}_{s,n}$. Therefore, when conditioning on $\mathsf{injective}^\Gamma_{\geq s}$, for any adversary $\mathcal{A}$ which only makes queries to $(f, \mathcal{O}, \mathcal{E}_{<s})$, the answers to $\mathcal{O}_{s,n}$ queries reveal exactly one point in the image of $\mathcal{O}_{s,n}$, while the answers to all other queries are independent of $\mathcal{O}_{s,n}$.

Recall that $\mathcal{O}_{s,n}$ is a function from $\{0,1\}^{2s}$ to $\{0,1\}^{10\ell(s,n)}$. Therefore, for any such $q'$-query $\mathcal{A}$, there are at most $2^{2s}$ points in the image of $\mathcal{O}_{s,n}$ that $\mathcal{A}$ can output which would cause $\mathsf{win}^\Gamma_{s,n}$ to occur, out of a total of at least $2^{10\ell(s,n)} - q'(s)$ points for $\mathcal{A}$ to choose from (because $\mathcal{A}$ could have learned at most $q'(s)$ points in the image of $\mathcal{O}_{s,n}$). Therefore,

$$\Pr_{\Gamma \leftarrow \mathfrak{S}_\ell, \mathcal{A}}\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] < \frac{2^{2s}}{2^{10\ell(s,n)} - q'(s)} < \frac{2^{2s}}{2^{10s} - q'(s)},$$

as desired. □

*Proof of Claim 5.14.* Let $p(s) = \frac{2^{2s} \cdot q(s)}{2^{10s} - 2q(s)^5}$ and suppose for contradiction there exists a $q$-query algorithm $M$ such that for infinitely many $s$, $\Pr_{\Gamma \leftarrow \mathfrak{S}_\ell}\left[\mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] > p(s)$. Let $p = p(s)$ and $q = q(s)$. We will show that $M$ can be used to construct an adversary $\mathcal{A}$ that makes $q' = 2q^5$ queries to $(f, \mathcal{O}, \mathcal{E}_{<s})$ and contradicts Lemma 5.15. Towards that end, let $\mathsf{win}^\Gamma_{s,n}$ be the event that $\mathcal{A}(1^s)$ succeeds at outputting a point in the image of $\mathcal{O}_{s,n}$ without receiving it from a query.

The adversary $\mathcal{A}$ does the following. First, sample $i^\star \leftarrow [q]$, and then run $M$ up until query $i^\star$, responding to all oracle queries as follows:

- For any query to $f$, $\mathcal{O}$, or $\mathcal{E}_{<s}$, query the real oracle and forward the corresponding answer. Store all query-answer pairs for $\mathcal{O}$ queries.
- For any query $\mathcal{E}_{s',n'}(\widetilde{C}, x)$ for $s' \geq s$, if $\widetilde{C}$ corresponds to a circuit $C$ from a previous $\mathcal{O}_{s',n'}$ query, evaluate $C^{\Gamma_{<s'}}(x)$ (responding to indirect queries as specified) and output the result. Otherwise, output $\perp$.

After $M$ makes the query with index $i^\star$, if it is an $\mathcal{E}_{s,n}$ query on some $(\widetilde{C}, x)$ where $\widetilde{C}$ was not in any previous $\mathcal{O}_{s,n}$ query, output $\widetilde{C}$ to the challenger. Otherwise, output $\perp$.

Observe that $\mathcal{A}$ makes at most $2q^5$ queries by Claim 5.10, because it makes all of $M$'s queries to $f$, $\mathcal{O}$, and $\mathcal{E}_{<s}$, and the remaining queries it makes are $M$'s indirect queries for some oracle in the support of $\mathfrak{S}_\ell$. We now analyze the success probability of $\mathcal{A}$. We have that

$$\begin{aligned}
\Pr\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] &= \Pr\left[\mathsf{win}^\Gamma_{s,n} \wedge \mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] + \Pr\left[\mathsf{win}^\Gamma_{s,n} \wedge \neg\mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] \\
&= \Pr\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{spoof}^\Gamma_{s,n} \wedge \mathsf{injective}^\Gamma_{\geq s}\right] \cdot \Pr\left[\mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] \\
&\geq \Pr\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{spoof}^\Gamma_{s,n} \wedge \mathsf{injective}^\Gamma_{\geq s}\right] \cdot p \\
&\geq \Pr\left[i^\star \text{ is the first instance of } \mathsf{spoof}^\Gamma_{s,n} \mid \mathsf{spoof}^\Gamma_{s,n} \wedge \mathsf{injective}^\Gamma_{\geq s}\right] \cdot p = p \cdot \frac{1}{q},
\end{aligned}$$

where the probability is over $\Gamma \leftarrow \mathfrak{S}_\ell$ and the random coins of $\mathcal{A}$. The first inequality is due to our assumption that $M$ causes $\mathsf{spoof}^\Gamma_{s,n}$ to occur with probability at least $p$. The second inequality is because the view of $M^{\mathcal{A}^\Gamma}$ is distributed exactly as in $M^\Gamma$ up until the first instance of $\mathsf{spoof}^\Gamma_{s,n}$. This is because all of $M$'s queries to $f$, $\mathcal{O}$, and $\mathcal{E}_{<s}$ are answered using the real oracle, and all queries $(\widetilde{C}, x)$ to $\mathcal{E}_{s',n'}$ for $s' \geq s$ are either answered by evaluating a pre-image of $\widetilde{C}$ under $\mathcal{O}_{s',n'}$, or with $\perp$. Since we are conditioning on $\mathsf{injective}^\Gamma_{\geq s}$, then if there is a pre-image of $\widetilde{C}$ from a previous query, it is the unique pre-image. Otherwise, since we are only considering queries before the first instance of $\mathsf{spoof}^\Gamma_{s,n}$, then any other queries to $\mathcal{E}_{s',n'}$ are answered with $\perp$, which is consistent with the distribution over $\Gamma$.

Therefore,

$$\Pr\left[\mathsf{win}^\Gamma_{s,n} \mid \mathsf{injective}^\Gamma_{\geq s}\right] \geq \frac{p}{q} = \frac{2^{2s} \cdot q}{q \cdot (2^{10s} - 2q^5)} = \frac{2^{2s}}{2^{10s} - 2q^5},$$

in contradiction with Lemma 5.15. □

### 5.3 Existence of a OWF Relative to $\Gamma$

**Theorem 5.16.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for some constant $0 \leq \epsilon < 1$. Then, given any oracle-aided $q$-query algorithm $\mathcal{A}$, it holds that for all $s \in \mathbb{N}$,*

$$\Pr_{x \leftarrow \{0,1\}^s, \Gamma \leftarrow \mathfrak{S}_\ell}\left[\mathcal{A}^\Gamma(f_s(x)) \in f_s^{-1}(f_s(x))\right] \leq \frac{q(s)^5}{2^{s-1}}$$

*In particular, this implies that for large enough $s$, if $q(s) < 2^{s/20}$, this probability is bounded by $2^{-s/2}$.*

**Proof.** The proof of this theorem is similar to that of [11], with the difference that we must emulate indirect $\mathcal{E}$ queries. Suppose for the sake of contradiction that there exists a PPT oracle-aided $q$-query adversary $\mathcal{A}$ that can invert $f_s$ with oracle access to $\Gamma$. We construct an adversary $\mathcal{B}$ that only makes queries to $\mathcal{O}$ and $f$, and simulates all queries to $\mathcal{E}$. The adversary $\mathcal{B}$ runs $\mathcal{A}(1^s)$ and responds to all oracle queries (including direct and indirect queries made by $\mathcal{A}$) as follows:

- For any query to $\mathcal{O}$ or $f$, the adversary $\mathcal{B}$ forwards the query to $\Gamma$ and returns the corresponding answer.

- For any query to $\mathcal{E}_{s',n'}$ for any $s'$ on $(\widetilde{C}, x)$, the adversary $\mathcal{B}$ enumerates over all pairs $(C, r) \in \{0,1\}^{2s;}$ in lexicographic order, queries $\mathcal{O}_{s',n'}(C, r)$, and checks if the response is $\widetilde{C}$. In the case that such a pre-image $(C, r)$ is found, $\mathcal{B}$ evaluates $C^{\Gamma<s'}(x)$ (responding to all oracle queries accordingly). Otherwise, if no pre-image is found, $\mathcal{B}$ returns $\perp$.

Observe that $\mathcal{B}$ simulates an oracle in the support of $\mathfrak{S}_\ell$, because for every $\mathcal{E}$ query, it finds the lexicographically first pre-image and evaluates it, just as done by the real oracle. The queries that $\mathcal{B}$ makes to $f_s$ fall into two categories—direct queries made by $\mathcal{A}$ to $f_s$, and queries to $f_s$ caused by indirect queries made by $\mathcal{A}$. Because $\mathcal{A}$ is a $q$-query algorithm, there are at most $q(s)$ direct queries to $f_s$ and by Claim 5.10, at most $q(s)^5$ indirect oracle queries. Therefore, the number of queries made by $\mathcal{B}$ is bounded by $2q(s)^5$. Since $f_s$ is a random function, any such algorithm $\mathcal{B}$ can output an inverse of $f_s(x)$ with probability at most $\frac{2q(s)^5}{2^s}$. □

### 5.4 Existence of XiO Relative to Γ

In this section we show that relative to Γ there exists an XiO scheme for the class $\mathcal{C}$ of all polynomial-size oracle-aided circuits. We proceed with the construction of the obfuscator.

**Construction 5.17.** *Let $\ell(s, n) = 2^{n\epsilon} \cdot s^2$ for a constant $0 \le \epsilon < 1$, and let $\Gamma \leftarrow \mathfrak{S}_\ell$. Then, for any class of oracle-aided circuits $\{\mathcal{C}_{s,n}\}_{s,n\in\mathbb{N}}$ relative to Γ, define $\mathsf{xi}\mathcal{O}^\Gamma = (\mathsf{Obf}^\Gamma, \mathsf{Eval}^\Gamma)$ as follows:*

- $\widetilde{C} \leftarrow \mathsf{Obf}^\Gamma(1^s, C)$*: On input $C \in \mathcal{C}_{s,n}$, sample $r \leftarrow \{0,1\}^s$ and query $\mathcal{O}_{s,n}(C, r)$ to obtain $\widehat{C}$. Then, enumerate over all inputs $x \in \{0,1\}^n$ and check that $\mathcal{E}_{s,n}(\widehat{C}, x) = C^{\Gamma<s}(x)$. If this holds for all $x$, output $(0, \widehat{C})$. Otherwise, output $(1, C)$, i.e., the original circuit (padded to size $10\ell(s, n)$).*[8]
- $y \leftarrow \mathsf{Eval}^\Gamma((b, \widetilde{C}), x)$*:* $\mathsf{Eval}$ *receives as input $(b, \widetilde{C}) \in \{0,1\} \times \{0,1\}^{10\ell(s,n)}$ and $x \in \{0,1\}^n$. If $b = 0$, then the algorithm* $\mathsf{Eval}$ *queries $\mathcal{E}_{s,n}(\widetilde{C}, x)$ and outputs the result. If $b = 1$, then* $\mathsf{Eval}$ *just evaluates $\widetilde{C}^{\Gamma<s}(x)$ and outputs the result.*

**Theorem 5.18.** *For any class of oracle-aided circuits $\mathcal{C}$, it holds that $\mathsf{xi}\mathcal{O}^\Gamma$ is a perfectly correct XiO for $\mathcal{C}$. Moreover, for any for any $q$-query adversary $\mathcal{D}$, if $q(s) < 2^{s/20}$, then*

$$\left|\Pr\left[\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) = 1\right] - \frac{1}{2}\right| \le 2^{-s/4}.$$

**Proof.** We show that $\mathsf{xi}\mathcal{O}$ satisfies perfect correctness, compression, and the indistinguishability requirement.

**Perfect correctness.** It is straightforward to verify that the $\mathsf{xi}\mathcal{O}$ construction achieves perfect correctness. Given a circuit $C$ as input, the algorithm $\mathsf{Obf}$ queries $\mathcal{O}$ to obtain $\widehat{C}$ and then enumerates over all inputs of the obfuscated circuit to see that the evaluation agrees with $C$. If the obfuscated circuit $\widehat{C}$ is perfectly correct, it is used as the obfuscation. Otherwise, the obfuscator outputs $C$, which trivially satisfies perfect correctness.

[8] We note that this technique, of enumerating all inputs, can only be done because we are constucting XiO. In particular, this step is the reason that this separation does not apply to perfectly correct SXiO.

**Compression.** We show that $\mathsf{xi}\mathcal{O}$ satisfies the efficiency required of XiO. For any $C \in \mathcal{C}_{s,n}$, it holds that

$$\mathsf{Outlen}\left[\mathsf{Obf}^{\Gamma}(1^s, C)\right] \leq \max\{s, 10\ell(s,n)\} + 1 = 2^{n\epsilon} \cdot s^2 + 1 \leq 2^{n\epsilon} \cdot \mathsf{poly}(s),$$

for some polynomial $\mathsf{poly}$.

With respect to the running time of the obfuscator on input $C$, it samples $r \leftarrow \{0,1\}^s$, makes a single $\mathcal{E}_{s,n}$ query to obtain $\widehat{C}$, and then for each input $x$, evaluates $\widehat{C}^{\Gamma_{<s}}(x)$. By Claim 5.10, it holds that $\widehat{C}^{\Gamma_{<s}}(x)$ can only cause $s^4$ indirect queries for each $x$. Moreover, the input and output lengths of each such query are bounded by $10\ell(s,n)$. Therefore, we have that

$$\mathsf{Time}\left[\mathsf{Obf}^{\Gamma}(1^s, C)\right] = s + \mathsf{poly}(2^n \cdot s^4 \cdot 10\ell(s,n)) \leq \mathsf{poly}(2^n, s)$$

for polynomials that depend on the oracle $\mathcal{O}$, thereby satisfying the compression of XiO.

**Security.** This proof is an adaptation of the proof in [11]. In particular, we have the following claim.

**Claim 5.19.** *For any for any $q$-query adversary $\mathcal{D}$, if $q(s) < 2^{s/20}$, then*

$$\left|\Pr\left[\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) = 1\right] - \frac{1}{2}\right| \leq 2^{-s/4}.$$

**Proof Sketch.** To show this, we follow the outline in [11]. Our oracle differs from theirs in three aspects. First, our oracle $\mathcal{O}$ is not necessarily injective, and [11] restrict $\mathcal{O}$ to be a length-increasing injective function. Second, the expansion factor of the oracle $\mathcal{O}$ is different from that of [11]. Third, we allow circuits $C$ to be oracle-aided with oracle access to $\Gamma_{<s}$, while [11] only allow circuits to have oracle access to $f$. We address these differences throughout the proof, and use results from [11] when relevant.

Let the challenge circuits be $C_0, C_1 \in \mathcal{C}_{s,n}$. The proof follows from a sequence of claims.

1. Suppose that there exists a $q$-query distinguisher $\mathcal{D}$ that wins $\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s)$ with probability at least $\frac{1}{2} + \delta$ for some $\delta > 0$. Let $q = q(s)$. It is easy to see that for such a distinguisher $\mathcal{D}$, if we focus on the case where $\mathcal{O}_{s',n'}$ is injective for all $s' \geq s$, it holds that

$$\Pr\left[\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) = 1 \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] \geq \frac{1}{2} + \delta - \Pr\left[\neg\mathsf{injective}^{\Gamma}_{\geq s}\right].$$

2. Given such a distinguisher $\mathcal{D}$, there exists a $(2q^5)$-query distinguisher $\mathcal{D}'$ that only makes oracle queries to $(f, \mathcal{O}, \mathcal{E}_{<s})$ with a related success probability. Let $\widetilde{\mathsf{Exp}}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D}',\mathcal{C}}(s)$ denote the experiment where the distinguisher $\mathcal{D}'$ does not make oracle calls to $\mathcal{E} \setminus \mathcal{E}_{<s}$. It follows from [11] that, conditioned on $\mathsf{injective}^{\Gamma}_{\geq s}$ and $\neg\mathsf{spoof}^{\mathcal{D},\ell}_{s,n}$, the advantage of $\mathcal{D}$ in $\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s)$ is equal to that of $\mathcal{D}'$ in $\widetilde{\mathsf{Exp}}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D}',\mathcal{C}}(s)$. In particular, by conditioning on $\mathsf{injective}^{\Gamma}_{\geq s}$, we construct the adversary $\mathcal{D}'$ just as in [11] with the difference that $\mathcal{D}'$ only has oracle access to $\mathcal{E}_{<s}$ (rather than $\mathcal{E}_{-s} = \left\{\mathcal{E}_{s',n'}\right\}_{s'<s,n'\leq s'}$, as in [11]) and must simulate indirect queries of $\mathcal{D}$ as well. Thus, by Claim 5.10, $\mathcal{D}'$ is a $q'$-query algorithm with $q'(s) < 2q(s)^5$. Therefore, this implies that

$$\begin{aligned}
&\Pr\left[\widetilde{\mathsf{Exp}}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D}',\mathcal{C}}(s) = 1 \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] \\
&\geq \Pr\left[\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s) = 1 \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] - \Pr\left[\mathsf{spoof}^{\mathcal{D},\ell}_{s,n} \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] \\
&\geq \frac{1}{2} + \delta - \Pr\left[\neg\mathsf{injective}^{\Gamma}_{\geq s}\right] - \Pr\left[\mathsf{spoof}^{\mathcal{D},\ell}_{s,n} \mid \mathsf{injective}^{\Gamma}_{\geq s}\right].
\end{aligned}$$

3. For $(b, r^*) \in \{0,1\} \times \{0,1\}^s$, we let $\widetilde{\mathsf{Exp}}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D}',\mathcal{C}}(s; b, r^*)$ denote the experiment in which the obfuscated circuit that the challenger delivers to $\mathcal{D}'$ is $\widetilde{C} = \mathcal{O}_{s,n}(C_b, r^*)$. We define the following two events:

   - Let $\mathsf{initialHit}_{s,n}$ be the event that $\mathcal{D}'$ makes an $\mathcal{O}_{s,n}$ query on either $(C_0, r^*)$ or $(C_1, r^*)$ prior to receiving the challenge circuit $\widetilde{C}$ from the challenger.
   - Let $\mathsf{hit}_{s,n}$ be the event that $\mathcal{D}'$ makes an $\mathcal{O}_{s,n}$ query on either $(C_0, r^*)$ or $(C_1, r^*)$ after receiving the challenge circuit $\widetilde{C}$ from the challenger.

   In [11], it is shown that for all $s, n \in \mathbb{N}$, the distinguishing probability of $\mathcal{D}'$ is exactly $1/2$ when both $\mathsf{initialHit}_{s,n}$ and $\mathsf{hit}_{s,n}$ do not occur. This applies to our case as well, because we are conditioning on $\mathsf{injective}^{\Gamma}_{\geq s}$, and there are no indirect queries that can reveal points in the image of $\mathcal{E}_{s,n}$ or $\mathcal{O}_{s,n}$. Therefore, the results of [11] imply that if $\mathcal{D}'$ succeeds to distinguish with probability greater than $1/2 + \delta'$ conditioned on $\mathsf{injective}^{\Gamma}_{\geq s}$, then the probability that $\mathsf{initialHit}_{s,n}$ or $\mathsf{hit}_{s,n}$ occur, conditioned on $\mathsf{injective}^{\Gamma}_{\geq s}$, is greater than $\delta'$.

4. On the other hand, [11] also showed that for every $q'$-query algorithm that does not make any queries to $\mathcal{E}_{s,n}$, it holds that

$$\Pr_{\substack{\mathcal{O} \\ (b,r^*)}} \left[\mathsf{initialHit}_{s,n} \vee \mathsf{hit}_{s,n}\right] < \frac{q'}{2^s - q'} \tag{4}$$

   which applies to our case when conditioning on $\mathsf{injective}^{\Gamma}_{\geq s}$ and because $\mathcal{D}'$ does not make any queries to $\mathcal{E} \setminus \mathcal{E}_{<s}$.

Putting everything together, assume towards a contradiction that there exists a $q$-query distinguisher $\mathcal{D}$ such that

$$\left| \Pr_{\substack{\mathcal{O} \\ (b,r^*)}} \left[\mathsf{Exp}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D},\mathcal{C}}(s; b, r^*)\right] - \frac{1}{2} \right| > \delta$$

for infinitely many $s \in \mathbb{N}$. This implies the existences of a $q'$-query algorithm $\mathcal{D}'$ with $q'(s) < 2q(s)^5$ that does not make any queries to $\mathcal{E} \setminus \mathcal{E}_{<s}$ for which

$$\begin{aligned}
&\left| \Pr_{\substack{\mathcal{O} \\ (b,r^*)}} \left[\widetilde{\mathsf{Exp}}^{XiO}_{\Gamma,\mathsf{xi}\mathcal{O},\mathcal{D}',\mathcal{C}}(s; b, r^*) \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] - \frac{1}{2} \right| \\
&\geq \frac{1}{2} + \delta - \Pr\left[\neg\mathsf{injective}^{\Gamma}_{\geq s}\right] - \Pr\left[\mathsf{spoof}^{\mathcal{D},\ell}_{s,n} \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] \\
&\geq \frac{1}{2} + \delta - \frac{1}{2^{5s-1}} - \frac{2^{2s} \cdot q}{2^{10s} - 2q^5} \geq \frac{1}{2} + \delta - \frac{1}{2^{4s}} - \frac{2^{2s} \cdot q}{2^{10s} - 2q^5}
\end{aligned}$$

for infinitely many $s \in \mathbb{N}$ and $n = n(s)$, by Claims 5.12 and 5.14. Taking $q < 2^{s/20}$, and $\delta = 2^{-s/4}$, we have that $q^5 < 2^{s/4}$ and thus the above implies that

$$\Pr_{\substack{\mathcal{O} \\ (b,r^*)}} \left[\mathsf{initialHit}_{s,n} \vee \mathsf{hit}_{s,n} \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] > \frac{1}{2^{s/4}} - \frac{1}{2^{4s}} - \frac{2^{2s} \cdot 2^{s/20}}{2^{10s} - 2^{s/4+1}} > \frac{1}{2^{s/4}} - \frac{1}{2^{4s}} - \frac{1}{2^{6s}} > \frac{1}{2^{s/4+1}} \ .$$

On the other hand, applying $q < 2^{s/20}$ in Eq. (4) with $q' = 2q^5$, we get that

$$\Pr_{\substack{\mathcal{O} \\ (b,r^*)}} \left[\mathsf{initialHit}_{s,n} \vee \mathsf{hit}_{s,n} \mid \mathsf{injective}^{\Gamma}_{\geq s}\right] < \frac{2q^5}{2^s - 2q^5} < \frac{2^{s/4+1}}{2^s - 2^{s/4+1}} = \frac{1}{2^{3s/4-1} - 1} \leq \frac{1}{2^{s/4+1}}$$

for $s \geq 5$. Since the above holds for infinitely many $s$, this is a contradiction. □

This completes the proof of Theorem 5.18. □

## 5.5 Breaking Perfect Key Agreement Relative to $\Gamma$

In this section, we will consider a key agreement protocol relative to $\Gamma$ between $\mathcal{A}$ and $\mathcal{B}$, and construct an adversary $E$ that breaks the key agreement protocol.

**Theorem 5.20.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for a constant $0 < \epsilon < 1$. Then, for any perfectly correct oracle-aided bit agreement protocol $\langle \mathcal{A}, \mathcal{B} \rangle$ in which $\mathcal{A}$ and $\mathcal{B}$ run in time at most most $q(s)$, there exists an oracle-aided adversary $E$ that makes $q(s)^{O(1)}$ oracle queries such that*

$$\left| \Pr\left[ \mathsf{Exp}^{\mathsf{KA}}_{\Gamma,(\mathcal{A},\mathcal{B}),E}(s) = 1 \right] - \frac{1}{2} \right| \geq \frac{7}{16} ,$$

*where the probability is over $\Gamma \leftarrow \mathfrak{S}_\ell$, and the randomness of $\mathcal{A}$, $\mathcal{B}$, and $E$. Moreover, the algorithm $E$ can be implemented in polynomial time given access to a* PSPACE*-complete oracle.*

**Proof.** Fix $\ell(s,n)$ as above and an execution of the key agreement protocol $\langle \mathcal{A}^\Gamma(1^s; r_A^\star), \mathcal{B}^\Gamma(1^s; r_B^\star) \rangle$. We start by defining some notation.

**Notation.** Let $Q_\mathcal{A}$, $Q_\mathcal{B}$, and $Q_E$ denote the set of oracle queries made by $\mathcal{A}$, $\mathcal{B}$, and $E$, respectively. Let $[O(x) = y] \in Q_p$ denote that a party $p$ queried an oracle $O$ on $x$ and received $y$. Let $Q_{\mathcal{AB}} = Q_\mathcal{A} \cup Q_\mathcal{B}$ be the set of oracle queries in the real protocol.

Since $\mathcal{A}$ and $\mathcal{B}$ are PPT algorithms, let $q = q(s)$ be a polynomial which upper bounds on the number of queries, size of each query-answer pair, and running time of $\mathcal{A}$ and $\mathcal{B}$. Thus, all $\mathcal{O}_{s,n}$ and $\mathcal{E}_{s,n}$ queries in the real execution of the protocol satisfy $s \leq q$ and $2^{n\epsilon} \cdot s^2 \leq q$. This implies that $n \leq \frac{1}{\epsilon} \log q$. We will use this bound on $n$ to show that $\mathcal{A}$ and $\mathcal{B}$ can only query $\mathcal{O}$ on circuits with logarithmic size input, and thus the adversary can learn the truth table of any circuit queried this way by only making a polynomial number of queries.

We now define an extended set of queries for any query-answer set $Q$. Intuitively, this captures queries that are "known" to an algorithm that makes the queries in $Q$. For example, suppose an algorithm $M$ queries $\mathcal{O}_{s,n}$ on some $(C, r)$ and obtains $\widetilde{C}$, and queries $\Gamma_{<s}$ on all queries in the evaluation of $C^{\Gamma_{<s}}(x)$ and learns that $C^{\Gamma_{<s}}(x) = y$. Then, intuitively $M$ knows that $\mathcal{E}_{s,n}(\widetilde{C}, x) = y$ (up to the probability of $\mathcal{O}$ being injective), even without making any $\mathcal{E}_{s,n}$ query. The following definition captures this dependence between the oracles.

**Definition 5.21.** *Given a query-answer set $Q$ and an oracle $\Gamma$, the* augmented query-answer set $\mathsf{Aug}(Q)$ *with respect to $\Gamma$ is defined recursively as follows:*

1. *Every query-answer pair in $Q$ is also in $\mathsf{Aug}(Q)$.*
2. *For every $\widetilde{C}$ such that there exists a query $[\mathcal{O}_{s,n}(C, r) = \widetilde{C}] \in \mathsf{Aug}(Q)$ or $[\mathcal{E}_{s,n}(\widetilde{C}, x) = y] \in \mathsf{Aug}(Q)$, the set $\mathsf{Aug}(Q)$ contains the following queries (and the corresponding answers):*
   (a) *$\mathcal{O}_{s,n}(C', r')$, if there exists a pair $(C', r')$ which is the lexicographically first such pair such that $\mathcal{O}_{s,n}(C', r') = \widetilde{C}$.*
   (b) *$\mathcal{E}_{s,n}(\widetilde{C}, w)$ for all $w \in \{0,1\}^n$.*
   (c) *All indirect queries made in the evaluation of $\mathcal{E}_{s,n}(\widetilde{C}, w)$ for all $w \in \{0,1\}^n$.*

We note that while the above definition is recursive, the set $\mathsf{Aug}(Q)$ is well-defined. In particular, for every $\widetilde{C}$ for which there is a related query in $\mathsf{Aug}(Q)$ to $\mathcal{O}_{s,n}$ or $\mathcal{E}_{s,n}$, the set adds one $\mathcal{O}_{s,n}$ query, $2^n$ queries to $\mathcal{E}_{s,n}$, and all indirect queries due to those queries. The $\mathcal{O}_{s,n}$ and $\mathcal{E}_{s,n}$ queries correspond to the same circuit, $\widetilde{C}$, and thus do not cause circularity, and the indirect queries must be for circuits of smaller sizes. We now bound the size of $\mathsf{Aug}(Q)$ for any query-answer set $Q$ made by a PPT algorithm. This will be helpful in bounding the number of queries that the adversary makes when simulating queries made by $\mathcal{A}$ and $\mathcal{B}$.

**Claim 5.22.** *Let $\ell(s,n) = 2^{n\epsilon} \cdot s^2$ for a constant $0 \leq \epsilon < 1$, and let $\Gamma \leftarrow \mathfrak{S}_\ell$. Let $M$ be a PPT oracle-aided algorithm relative to $\Gamma$, and let $Q$ be the set of queries made by $M(1^s)$. Then, there exists a polynomial* poly *such that* $|\mathsf{Aug}(Q)| = \mathsf{poly}(s)$.

**Proof.** Let $q = q(s)$ be a polynomial upper bound on the size of the queries and answers for all queries made by $M$. Without loss of generality, assume that $Q$ contains only one query (if $Q$ contains $k$ queries, the resulting bound will have an additional multiplicative factor of $k$). For any value $\widetilde{C}$, let $\widetilde{C}$ be *associated* with $Q$ if there is a query $\mathcal{O}_{s,n}(C,r) = \widetilde{C}$ or $\mathcal{E}_{s,n}(\widetilde{C},x) = y$ in $Q$. Moreover, for any $\widetilde{C}$ associated with $Q$, let $\mathsf{Extend}(\widetilde{C})$ denote the set of queries added in one iteration of Step 2 of Definition 5.21 due to $\widetilde{C}$. With this notation, the process to form $\mathsf{Aug}(Q)$ from $Q$ can be written as:

> While there exists a value $\widetilde{C}$ associated $\mathsf{Aug}(Q)$ such that $\mathsf{Extend}(\widetilde{C}) \not\subseteq \mathsf{Aug}(Q)$, add $\mathsf{Extend}(\widetilde{C})$ to $\mathsf{Aug}(Q)$.

To prove the claim, we first bound the size of $\mathsf{Extend}(\widetilde{C})$ for any $\widetilde{C}$ associated with $\mathsf{Aug}(Q)$, and then we bound the number of such associated values. To bound $|\mathsf{Extend}(\widetilde{C})|$, observe that for any $\widetilde{C}$ associated with $Q$, by definition there is a query $\mathcal{O}_{s,n}(C,r) = \widetilde{C}$ or $\mathcal{E}_{s,n}(\widetilde{C},x) = y$ in $Q$. Both of these queries have size greater than $\ell(s,n) = 2^{n\epsilon} \cdot s^2$. Therefore, the size of each query is bounded above by $q$, it holds that $s < q^{\frac{1}{2}}$ and $n < \frac{1}{\epsilon}\log(q)$. By definition of the augmented set, this extends to any $\widetilde{C}$ associated with $\mathsf{Aug}(Q)$. Moreover, for any $\widetilde{C} \in \{0,1\}^{10\ell(s,n)}$ associated with $\mathsf{Aug}(Q)$, the set $\mathsf{Extend}(\widetilde{C})$ contains at most 1 query to $\mathcal{O}_{s,n}$, $2^n$ queries to $\mathcal{E}_{s,n}$, and at most $2^n \cdot s^5$ queries to $\Gamma_{<s}$, because each query to $\mathcal{E}_{s,n}$ can cause at most $s^5$ indirect queries by Claim 5.10. Therefore, we have that

$$|\mathsf{Extend}(\widetilde{C})| \leq 1 + 2^n + 2^n \cdot s^5 \leq 1 + 2^{\frac{1}{\epsilon}\log(q)} + 2^{\frac{1}{\epsilon}\log(q)} \cdot q^{\frac{5}{2}} \leq 3q^{\frac{5}{2}+\frac{1}{\epsilon}} = 3q^d$$

for a constant $d = \frac{5}{2} + \frac{1}{\epsilon}$.

We now turn to bound the number of values $\widetilde{C}$ associated with $\mathsf{Aug}(Q)$. Towards this end, observe that for any $\widetilde{C}$, the set $\mathsf{Extend}(\widetilde{C})$ can be partioned into two sets $A$ and $B$ as follows:

- $A$ contains all queries to $\mathcal{O}_{s,n}$, $\mathcal{E}_{s,n}$, and $f$.
- $B$ contains all queries to $\mathcal{O}_{<s}$ and $\mathcal{E}_{<s}$. In particular, since $B$ contains all indirect queries due to queries to $\mathcal{E}_{s,n}$, any query in $B$ must be to $\mathcal{O}_{s',n'}$ or $\mathcal{E}_{s',n'}$ for some $s' \leq s^{\frac{1}{2}}$.

Using this notation, we have that $\mathsf{Aug}(Q)$ can be written as $A \cup \mathsf{Aug}(B)$, because the queries in $A$ are those that do not cause new values to be associated with $\mathsf{Aug}(Q)$, while the queries in $B$ are those that cause further recursion. Moreover, the queries in $B$ all have size bounded by $s^{\frac{1}{2}} \leq q^{\frac{1}{2}}$. Therefore, letting $T(q)$ denote an upper bound on the size of the augmented set for any set of queries with query size bounded by $q$, we have that

$$T(q) \leq |A| + |B| \cdot T(q^{\frac{1}{2}}) \leq 3q^d + 3q^d \cdot T(q^{\frac{1}{2}}),$$

where we used the fact that both $A$ and $B$ have size bounded by $|\mathsf{Extend}(\widetilde{C})| \leq 3q^d$. Noting that there can be at most $\log\log(q)$ levels of recursion and enumerating shows that

$$T(q) \leq \sum_{i=0}^{\log\log(q)} 3^{i+1} \cdot q^{\sum_{j=0}^{i} \frac{d}{2^j}} \leq 3^{\log\log(q)+1} \sum_{i=0}^{\log\log(q)} \cdot q^{\sum_{j=0}^{i} \frac{d}{2^j}} < 3\log^2(q) \cdot \sum_{i=0}^{\log\log(q)} q^{d\sum_{j=0}^{i} \frac{1}{2^j}}$$
$$< 3\log^2(q) \cdot \sum_{i=0}^{\log\log(q)} q^{2d} = 3\log^2(q) \cdot (\log\log(q)+1) \cdot q^{2d} = q^{O(1)}.$$

Therefore, $|\mathsf{Aug}(Q)|$ is polynomial in $q$ and thus polynomial in $s$, as desired. □

Let $q' = q'(s)$ denote the polynomial upper bound on $|\mathsf{Aug}(Q_{\mathcal{AB}})|$ computed in Claim 5.22. We are now ready to define the adversary $E$.

**The adversary $E$.**

- **Input:** A transcript $T$ of an execution $\langle \mathcal{A}^{\Gamma}(1^s; r_A^\star), \mathcal{B}^{\Gamma}(1^s; r_B^\star)\rangle$.
- **Oracle Access:** $\Gamma = (f, \mathcal{O}, \mathcal{E})$.
- **Algorithm:**
  1. Initialize a set $Q_E = \emptyset$ and a multiset $K = \emptyset$.
  2. **Learning $\mathsf{Im}(\Gamma_s)$ for small $s$:** For every $s$ such that $2^{10s} - 2^{2s} < 2q'$, query $\Gamma_s$ on all inputs and add these query-answer pairs to $Q_E$.
  3. Repeat the following $2q' + 1$ times:
     (a) **Simulation phase:** Find a valid oracle $\Gamma' = (f', \mathcal{O}', \mathcal{E}')$ and random strings $r'_A, r'_B$ such that the following holds:
        i. Every query in $Q_E$ is answered the same way in $\Gamma'$ as in $Q_E$.
        ii. $\mathcal{O}'_{s,n}$ is injective for all $s, n \in \mathbb{N}$.
        iii. The transcript $T'$ outputted by $\langle \mathcal{A}^{\Gamma'}(1^s; r'_A), \mathcal{B}^{\Gamma'}(1^s, r'_B)\rangle$ is the same as $T$.

        Abort if no such $\Gamma', r'_A, r'_B$ exist. Let $k'_A$ be the key outputted by $\mathcal{A}$ in this simulation, and add $k'_A$ to $K$.
     (b) **Update phase:** Let $Q_{\mathsf{Sim}}$ be the queries made by $\mathcal{A}$ and $\mathcal{B}$ in the execution $\langle \mathcal{A}^{\Gamma'}(1^s; r'_A), \mathcal{B}^{\Gamma'}(1^s, r'_B)\rangle$, and consider the set $\mathsf{Aug}(Q_{\mathrm{Sim}})$ with respect to $\Gamma'$. Query $\Gamma$ with all queries in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \setminus Q_E$ and update $Q_E$ with these queries and answers.
- **Output:** The majority key $k$ in $K$.

**Lemma 5.23.** *$E$ makes $\mathsf{poly}(q)$ many queries some polynomial $\mathsf{poly}$.*

**Proof.** $E$ first queries $\Gamma_s$ on all inputs for small integers $s'$ satisfying $2^{10s'} - 2^{2s'} < 2q'$. Since $q'$ is polynomial in $q$, this can be done by querying $\mathcal{O}_{s',n'}$ on all inputs for $s' \in O(\log(q))$, thus resulting in polynomially many queries in $q$. $E$ then makes at most $|\mathsf{Aug}(Q_{\mathrm{Sim}})|$ queries in the update phase in each iteration. Since $|Q_{\mathsf{Sim}}| < q$, it holds that $|\mathsf{Aug}(Q_{\mathrm{Sim}})|$ is polynomial in $q$. Moreover, $E$ runs for $2q' + 1$ iterations, which is polynomial in $q$, thus proving the lemma. □

**Bad event.** We will show that the adversary $E$ always succeeds to find the key computed in the real protocol, assuming that $\mathcal{O}$ is an injective function. We define $\mathsf{injective}^{\Gamma}_{s,n}$ and $\mathsf{injective}^{\Gamma}$ as in Definition 5.11. By Claim 5.12, we have that

$$\Pr\left[\neg\mathsf{injective}^{\Gamma}\right] \leq \sum_{s=1}^{\infty} \frac{1}{2^{6s}} \leq 2^{-4}.$$

We proceed to our main lemma.

**Lemma 5.24.** *Let $k^\star$ denote the key computed by $\mathcal{A}$ and $\mathcal{B}$ in the real execution of the protocol. If $\mathsf{injective}^{\Gamma}$ holds, then $E$ does not abort, and in each iteration either (1) $E$ adds a query in $\mathsf{Aug}(Q_{\mathcal{AB}})$ to $Q_E$, or (2) $E$ adds $k^\star$ to $K$.*

**Proof.** We first show that assuming $\mathsf{injective}^{\Gamma}$ holds, $E$ does not abort. Recall that $E$ aborts if it cannot find a valid oracle $\Gamma'$ and strings $r'_{\mathcal{A}}, r'_{\mathcal{B}}$ such that $\Gamma'$ is consistent with $Q_E$, the oracle $\mathcal{O}'_{s,n}$ is injective, and the transcript $T'$ outputted by the simulated execution with respect to $\Gamma', r'_{\mathcal{A}}$ and $r'_{\mathcal{B}}$ is the same as the real transcript $T$. As the real oracle $\Gamma$ and the real randomness $r^\star_{\mathcal{A}}$, $r^\star_{\mathcal{B}}$ satisfy these properties, there exists at least one valid oracle and pair of random strings and therefore $E$ does not abort.

We now show that in every iteration, either (1) $E$ adds a query in $\mathsf{Aug}(Q_{\mathcal{AB}})$ to $Q_E$, or (2) $E$ adds $k^\star$ to $K$. Consider some iteration in which (1) does not hold. We will show that $E$ adds $k^\star$ to $K$ in this iteration. Let $\Gamma', r'_{\mathcal{A}}, r'_{\mathcal{B}}$ be the oracle and random strings chosen by $E$ in this iteration. By definition, the transcript of this execution is $T$. Let $k'$ be the key outputted by $\langle \mathcal{A}^{\Gamma'}(1^s; r'_{\mathcal{A}}), \mathcal{B}^{\Gamma'}(1^s; r'_{\mathcal{B}})\rangle$. Assuming that (1) does not hold, we now show that there exists a hybrid oracle $\widetilde{\Gamma}$ for which

$$(k', k^\star, T) \leftarrow \langle \mathcal{A}^{\widetilde{\Gamma}}(1^s; r'_{\mathcal{A}}), \mathcal{B}^{\widetilde{\Gamma}}(1^s; r^\star_{\mathcal{B}})\rangle.$$

That is, we show an oracle $\widetilde{\Gamma}$ such that when $\mathcal{A}$ uses the randomness of the simulation and $\mathcal{B}$ uses the randomness of the real protocol and both run with respect to $\widetilde{\Gamma}$, $\mathcal{A}$ outputs $k'$ (as in the simulation) while $\mathcal{B}$ outputs $k^\star$ (as in the real), and the execution produces the transcript $T$ (as in both the real and simulated protocols). Given the existence of such an oracle, by the perfect correctness, it must hold that $k' = k^\star$, and therefore, since $E$ adds $k' = k^\star$ to $K$, the claim follows.

We construct the hybrid oracle $\widetilde{\Gamma} = (\widetilde{f}, \widetilde{\mathcal{O}}, \widetilde{\mathcal{E}})$ as follows:

- **The oracle $\widetilde{f}$.** For every $s$, for every $x$ such that $[f'_s(x) = y] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$, set $\widetilde{f}_s(x) = y$. For every $x$ such that $[f_s(x) = y] \in \mathsf{Aug}(Q_{\mathcal{AB}})$, set $\widetilde{f}_s(x) = y$. For every other $x$, set $\widetilde{f}_s(x) = 0$.
- **The oracle $\widetilde{\mathcal{O}}$.** For every $s$ and $n$, proceed as follows. For every $(C, r) \in \{0,1\}^{2s}$ for which $[\mathcal{O}'_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$, set $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widehat{C}$. Likewise, for every $(C, r) \in \{0,1\}^{2s}$ for which $[\mathcal{O}_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$, set $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widetilde{C}$. For every other $(C, r) \in \{0,1\}^{2s}$ for which $\widetilde{\mathcal{O}}_{s,n}$ is not yet defined, set the value $\mathcal{O}_{s,n}(C, r)$ arbitrarily, such that it avoids the set $\mathsf{avoid}_{s,n}$, defined as

$$\begin{aligned}\mathsf{avoid}_{s,n} \stackrel{\mathsf{def}}{=} &\left\{\widetilde{C} : [\mathcal{O}_{s,n}(\star, \star) = \widetilde{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}}) \text{ or } [\mathcal{E}_{s,n}(\widetilde{C}, \star) = \star] \in \mathsf{Aug}(Q_{\mathcal{AB}})\right\} \\ &\cup \left\{\widetilde{C} : [\mathcal{O}'_{s,n}(\star, \star) = \widetilde{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}}) \text{ or } [\mathcal{E}'_{s,n}(\widetilde{C}, \star) = \star] \in \mathsf{Aug}(Q_{\mathrm{Sim}})\right\}\end{aligned}$$

  where $\star$ represents an arbitrary value (that may be $\perp$). In particlar, the set $\mathsf{avoid}_{s,n}$ will ensure that for any string $\widetilde{C} \in \{0,1\}^{10\ell(s,n)}$ that is associated with $\mathsf{Aug}(Q_{\mathcal{AB}})$ or $\mathsf{Aug}(Q_{\mathrm{Sim}})$, there will not be a pre-image of $\widetilde{C}$ under $\widetilde{\mathcal{O}}$ other than the one specified by $\mathsf{Aug}(Q_{\mathcal{AB}})$ or $\mathsf{Aug}(Q_{\mathrm{Sim}})$. This helps us show that there are no conflicting evaluations under $\widetilde{\mathcal{E}}_{s,n}$. Moreover, note that $\mathsf{avoid}_{s,n}$ has size at most $|\mathsf{Aug}(Q_{\mathcal{AB}})| + |\mathsf{Aug}(Q_{\mathrm{Sim}})| \leq 2q'$, while $\mathcal{O}_{s,n}$ has a domain of size $2^{2s}$ and a range of size $2^{10\ell(s,n)} > 2^{10s}$. Note that for any $s$ such that $2^{10s} - 2q' < 2^{2s}$, all $C$ already have images under $\widetilde{O}$ because $E$ queries $\Gamma_s$ on all queries for these $s$. Thus, for any $s$ such that an arbitrary image of $(C, r)$ is chosen under $\mathcal{O}_{s,n}$, there are enough strings such $(C, r)$ will have an image under $\mathcal{O}_{s,n}$.
- **The oracle $\widetilde{\mathcal{E}}$.** This oracle is defined iteratively. For each $s \in \mathbb{N}$ and each $n \leq s$, define $\widetilde{\mathcal{E}}_{s,n}$ deterministically based on $\widetilde{f}_{<s}$, $\widetilde{\mathcal{O}}_{<s}$ and $\widetilde{\mathcal{E}}_{<s}$, exactly as $\mathcal{E}_{s,n}$ is defined with respect to $\Gamma_{<s}$.

We now analyze an execution of the protocol with respect to the oracle $\widetilde{\Gamma}$ while $\mathcal{A}$ uses the randomness $r'_{\mathcal{A}}$ (as in the simulation) and $\mathcal{B}$ uses randomness $r^\star_{\mathcal{B}}$ (as in the real). Let

$$(\widetilde{k}_A, \widetilde{k}_B, \widetilde{T}) \leftarrow \langle \mathcal{A}^{\widetilde{\Gamma}}(1^s; r'_A), \mathcal{B}^{\widetilde{\Gamma}}(1^s; r^\star_B)\rangle.$$

We will show that $\widetilde{T} = T$, $\widetilde{k}_A = k'$, and $\widetilde{k}_B = k^\star$. Towards this end, it is enough to show that $\widetilde{\Gamma}$ agrees with $\Gamma$ on all queries in $\mathsf{Aug}(Q_{\mathcal{AB}})$ and that $\widetilde{\Gamma}$ agrees with $\Gamma'$ on all queries in $\mathsf{Aug}(Q_{\mathrm{Sim}})$. Since $E$ adds all queries in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \setminus Q_E$ to $Q_E$ in each round, and we assumed that $E$ does not add any queries in $\mathsf{Aug}(Q_{\mathcal{AB}})$ to $Q_E$ in this iteration, it implies that all query and answer pairs in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \cap \mathsf{Aug}(Q_{\mathcal{AB}})$ agree with the real oracle $\Gamma$. As a result, it is enough to show that all queries in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \cup \mathsf{Aug}(Q_{\mathcal{AB}})$ are answered the same in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \cup \mathsf{Aug}(Q_{\mathcal{AB}})$ as in $\widetilde{\Gamma}$.

First, it is easy to see that $\widetilde{f}$ is consistent with all queries to $f$ and $f'$ in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$ because there are no contradicting queries between $\mathsf{Aug}(Q_{\mathcal{AB}})$ and $\mathsf{Aug}(Q_{\mathrm{Sim}})$. Similarly, $\widetilde{\mathcal{O}}$ is consistent with all queries to $\mathcal{O}$ and $\mathcal{O}'$ in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$.

As for $\widetilde{\mathcal{E}}$, to show that $\widetilde{\mathcal{E}}$ is consistent with $\mathcal{E}$ and $\mathcal{E}'$ queries in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$, we show the following stronger statement by induction on $s$. For every $s \in \mathbb{N}$, the following holds:

**(a)** For every $n \leq s$, consider any query on $(C, r)$ to $\mathcal{O}_{s,n}$ or $\mathcal{O}'_{s,n}$ in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$. If there exists a $(C', r')$ such that $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widetilde{\mathcal{O}}_{s,n}(C', r')$, then $C$ and $C'$ are functionally equivalent with respect to $\widetilde{\Gamma}$.

**(b)** Assuming **(a)** holds for $s$, then for any $n \leq s$, any $\mathcal{E}_{s,n}$ or $\mathcal{E}'_{s,n}$ query that appears in $\mathsf{Aug}(Q_{\mathrm{Sim}}) \cup \mathsf{Aug}(Q_{\mathcal{AB}})$ is answered the same by $\widetilde{\mathcal{E}}_{s,n}$.

We show **(a)** for $\mathcal{O}$ queries in $\mathsf{Aug}(Q_{\mathcal{AB}})$ and **(b)** for $\mathcal{E}$ queries in $\mathsf{Aug}(Q_{\mathcal{AB}})$. The cases of $\mathcal{O}'$ or $\mathcal{E}'$ queries in $\mathsf{Aug}(Q_{\mathrm{Sim}})$ follow analogously.

**Base case of (a).** Consider any query $[\mathcal{O}_{1,1}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$. The existence of this query implies that $\widetilde{\mathcal{O}}_{1,1}(C, r) = \widehat{C}$. Suppose there is a pair $(C', r') \neq (C, r)$ such that $\widetilde{\mathcal{O}}_{1,1}(C, r) = \widetilde{\mathcal{O}}_{1,1}(C', r') = \widehat{C}$. We want to show that $C$ is functionally equivalent to $C'$ relative to $\widetilde{\Gamma}$. Note that $C$ and $C'$ have no oracle gates due to their size, so we only need to show that that $C$ and $C'$ are functionally equivalent.

Recall that $\widetilde{\mathcal{O}}$ "inherits" all query-answer pairs from queries to $\mathcal{O}$ and $\mathcal{O}'$ in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$, and chooses arbitrary images for inputs that are independent of $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$. Thus, the only way that there exists a $(C', r') \neq (C, r)$ such that $\widetilde{\mathcal{O}}_{1,1}(C', r') = \widehat{C}$ is if there is a query $[\mathcal{O}'_{1,1}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$. In particular, $\mathcal{O}_{1,1}(C', r')$ cannot result in $\widehat{C}$ because $\mathcal{O}_{1,1}$ is injective, and $\widehat{C}$ cannot be chosen as an arbitrary image of $C', r'$ under $\widetilde{\mathcal{O}}$ because it is in $\mathsf{avoid}_{1,1}$.

Therefore, there exist queries $[\mathcal{O}_{1,1}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ and $[\mathcal{O}'_{1,1}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$, and our goal is to show that $C$ and $C'$ are functionally equivalent. Because $\mathcal{O}'_{1,1}$ and $\mathcal{O}_{1,1}$ are injective, by definition of $\mathsf{Aug}(Q_{\mathcal{AB}})$ and $\mathsf{Aug}(Q_{\mathrm{Sim}})$, there exist queries $[\mathcal{E}'_{1,1}(\widehat{C}, x) = C'(x)] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$ and $[\mathcal{E}_{1,1}(\widehat{C}, x) = C(x)] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ for every $x \in \{0, 1\}$. Since there are no contradicting queries in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$, this implies that $C(x) = C'(x)$ for every $x \in \{0, 1\}$. Therefore, $C$ and $C'$ are functionally equivalent.

**Base case of (b).** Here, we show that any $\mathcal{E}_{1,1}$ query that appears in $\mathsf{Aug}(Q_{\mathcal{AB}})$ is answered the same by $\widetilde{\mathcal{E}}_{1,1}$. There are two cases to consider.

- $[\boldsymbol{\mathcal{E}_{1,1}(\widehat{C}, x) = y}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ **with** $\boldsymbol{y \neq \perp}$**.** In this case, because $\mathcal{O}$ is injective, there exists a unique pair $(C, r)$ such that $[\mathcal{O}_{1,1}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ with $C(x) = y$. Now, consider the hybrid oracle $\widetilde{\mathcal{E}}_{1,1}$. On input $(\widehat{C}, x)$, the oracle $\widetilde{\mathcal{E}}_{1,1}$ finds the lexicographically first pair $(C', r')$ with $\widetilde{\mathcal{O}}_{1,1}(C', r') = \widehat{C}$. Here, since $[\mathcal{O}_{1,1}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$, it implies that $\widetilde{\mathcal{O}}_{1,1}(C, r) = \widehat{C}$, so we can apply the base case of **(a)** which gives us that $C$ is functionally equivalent to $C'$. Therefore, $\widetilde{\mathcal{E}}_{1,1}(\widehat{C}, x) = C'(x) = C(x) = y$, as desired.

- $[\boldsymbol{\mathcal{E}_{1,1}(\widehat{C}, x) = \perp}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$**.** In this case, observe that any pre-image of $\widehat{C}$ under $\mathcal{O}_{1,1}$ would be too small to have oracle gates. Thus, if an $\mathcal{E}_{1,1}$ query on $(\widehat{C}, x)$ returns $\perp$, it must be that $\widehat{C}$ is not in the image of $\mathcal{O}_{1,1}$. This also holds with respect to the simulated and hybrid oracles.

  Therefore, since our goal is to show that $\widetilde{\mathcal{E}}_{1,1}(\widehat{C}, x) = \perp$, it is enough to show that $\widehat{C}$ is not in the image of $\widetilde{\mathcal{O}}_{1,1}$. Suppose for contradiction that $\widehat{C}$ is in the image of $\widetilde{\mathcal{O}}_{1,1}$. By construction of $\widetilde{\mathcal{O}}_{1,1}$, it must be that there exists a $(C, r)$ such that $[\mathcal{O}'_{1,1}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$. By

definition of the augmented set and the fact that $\mathcal{O}'_{1,1}$ is injective, this implies the existence of a query $[\mathcal{E}'_{1,1}(\widehat{C}, x) = C(x)] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$. Note that $C$ is too small to have oracle gates and thus $C(x) \neq \perp$. However, this implies that $E$ learns a new query in $\mathsf{Aug}(Q_{\mathcal{AB}})$ during the update phase, in contradiction.

We now show the inductive step. Suppose that **(a)** and **(b)** hold for all $s'$ with $s' < s$.

**Inductive step for (a).** We now show that **(a)** holds for $s$. Fix any $n \leq s$ and consider any query $[\mathcal{O}_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$. This implies that $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widehat{C}$. Suppose that there exists a pair $(C', r') \neq (C, r)$ with $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widetilde{\mathcal{O}}_{s,n}(C', r') = \widehat{C}$. Our goal is to show that $C$ and $C'$ are functionally equivalent under $\widetilde{\Gamma}$. By the same logic as the base case of **(a)**, by construction of $\widetilde{\Gamma}$, the only way that such a $(C', r')$ exists is if there is a query $[\mathcal{O}'_{s,n}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$.

Therefore, there exists queries $[\mathcal{O}_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ and $[\mathcal{O}'_{s,n}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$. Because both $\mathcal{O}_{s,n}$ and $\mathcal{O}'_{s,n}$ are injective, these queries imply that $(C, r)$ and $(C', r')$ are the unique pre-images of $\widehat{C}$ under $\mathcal{O}_{s,n}$ and $\mathcal{O}'_{s,n}$, respectively. Therefore, there exist queries $[\mathcal{E}_{s,n}(\widehat{C}, x) = C^{\Gamma_{<s}}(x)] \in \mathsf{Aug}(Q_{\mathcal{AB}})$ and $[\mathcal{E}'_{s,n}(\widehat{C}, x) = C'^{\Gamma'_{<s}}(x)] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$ for all $x \in \{0,1\}^n$ (where the evaluations of $C^{\Gamma_{<s}}$ and $C'^{\Gamma'_{<s}}$ may be $\perp$), by definition of the augmented sets.

Recall that we want to show that $C$ is functionally equivalent to $C'$ under $\widetilde{\Gamma}$. This amounts to showing that $C^{\widetilde{\Gamma}_{<s}}(x) = C'^{\widetilde{\Gamma}_{<s}}(x)$ for every $x$. Therefore, fix any $x \in \{0,1\}^n$. Because there can be no contradicting queries between $\mathsf{Aug}(Q_{\mathcal{AB}})$ and $\mathsf{Aug}(Q_{\mathrm{Sim}})$, the existence of the $\mathcal{E}$ and $\mathcal{E}'$ queries mentioned above imply that $C^{\Gamma_{<s}}(x) = C'^{\Gamma'_{<s}}(x)$. All indirect queries by $C$ and $C'$ have sizes smaller than $s$. Moreover, all indirect queries made by $C^{\Gamma_{<s}}(x)$ appear in $\mathsf{Aug}(Q_{\mathcal{AB}})$, and all indirect queries made by $C'^{\Gamma'_{<s}}(x)$ appear in $\mathsf{Aug}(Q_{\mathrm{Sim}})$. Therefore, by part **(b)** of the inductive hypothesis, $\widetilde{\Gamma}_{<s}$ agrees with $\Gamma_{<s}$ and $\Gamma'_{<s}$ on each of these queries. Therefore,

$$C^{\widetilde{\Gamma}_{<s}}(x) = C^{\Gamma_{<s}}(x) = C'^{\Gamma'_{<s}}(x) = C'^{\widetilde{\Gamma}_{<s}}(x),$$

so $C$ and $C'$ are functionally equivalent under $\widetilde{\Gamma}$.

**Inductive step for (b).** Assuming **(a)** holds for $s$, we now show that **(b)** holds for $s$. Let $n \leq s$ and suppose there is a query $[\mathcal{E}_{s,n}(\widehat{C}, x) = y] \in \mathsf{Aug}(Q_{\mathcal{AB}})$, where $y$ may be $\perp$. We want to show that $\widehat{\mathcal{E}}_{s,n}(\widehat{C}, x) = y$, that is, the hybrid oracle agrees with the given query. There are two cases to consider.

- There exists a query $[\mathcal{O}_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$. Then, because $\mathcal{O}$ is injective, it holds that $(C, r)$ is the unique pre-image of $\widehat{C}$ under $\mathcal{O}_{s,n}$. Thus, $\mathcal{E}_{s,n}(\widehat{C}, x)$ evaluates $C^{\Gamma_{<s}}(x)$ to obtain $y$. Now, consider the hybrid oracle $\widetilde{\mathcal{E}}_{s,n}(\widehat{C}, x)$, which looks for the lexicographically first pre-image of $\widehat{C}$ under $\widetilde{\mathcal{O}}_{s,n}$. Because there are no contradicting queries in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$, we know that $\widetilde{\mathcal{O}}_{s,n}(C, r) = \widehat{C}$, but $(C, r)$ may not be the lexicographically first pair for which this holds. Nevertheless, we can apply part **(a)** of the inductive hypothesis for $s$ to show that $\widetilde{\mathcal{E}}_{s,n}(\widehat{C}, x) = C^{\widetilde{\Gamma}_{<s}}(x)$. Since $\widetilde{\Gamma}_{<s}$ agrees with $\Gamma_{<s}$ on all queries in the evaluation of $C^{\widetilde{\Gamma}_{<s}}(x)$ by part **(b)** of the inductive hypothesis, it holds that $C^{\widetilde{\Gamma}_{<s}}(x) = C^{\Gamma_{<s}}(x) = y$. Therefore, $\widetilde{\mathcal{E}}_{s,n}(\widehat{C}, x) = y$ as desired.
- There is no $(C, r)$ such that $[\mathcal{O}_{s,n}(C, r) = \widehat{C}] \in \mathsf{Aug}(Q_{\mathcal{AB}})$. In this case, it must be that there is no pre-image of $\widehat{C}$ under $\mathcal{O}_{s,n}$, so $y = \perp$. Thus, we want to show that $\widetilde{\mathcal{E}}_{s,n}(\widehat{C}, x) = \perp$. If $\widehat{C}$ is not in the image of $\widetilde{\mathcal{O}}_{s,n}$, it directly implies that $\widetilde{\mathcal{E}}_{s,n}(\widehat{C}, x) = \perp$, so we focus on the case where $\widehat{C}$ is in the image of $\widetilde{\mathcal{O}}_{s,n}$. This case could only only if there exists a pair $(C', r')$ such that $[\mathcal{O}'_{s,n}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$, by construction of $\widetilde{\mathcal{O}}_{s,n}$.

We show that this is analogous to the first case of the inductive step. The query $[\mathcal{O}'_{s,n}(C', r') = \widehat{C}] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$ implies the existence of the query $[\mathcal{E}'_{s,n}(\widehat{C}, x) = \bot] \in \mathsf{Aug}(Q_{\mathrm{Sim}})$, because if this query did not result in $\bot$, there would be a contradicting query in $\mathsf{Aug}(Q_{\mathcal{AB}}) \cup \mathsf{Aug}(Q_{\mathrm{Sim}})$. Thus, we can apply the same logic as the first case, replacing queries to $\Gamma$ with those to $\Gamma'$ and replacing $y$ with $\bot$, which completes the proof.

We reiterate that the cases for **(a)** and **(b)** corresponding to queries in $\mathsf{Aug}(Q_{\mathrm{Sim}})$ rather than $\mathsf{Aug}(Q_{\mathcal{AB}})$ are analogous. This completes the proof of Lemma 5.24. □

**Wrapping up.** Given a perfectly-correct key agreement protocol $\langle \mathcal{A}, \mathcal{B} \rangle$ bounded by some running time $q(s)$, we showed the existence of an adversary $E$ that makes $q^{O(1)}$ queries and finds the key $k^\star$ with probability at least $1 - 2^{-4}$. Because $q(\cdot)$ is a polynomial, we conclude that $E$ makes at most polynomial number of oracle queries to $\Gamma$. Moreover, all other computations that are done by $E$ can be done using a polynomial number of queries to a $\mathsf{PSPACE}$-complete oracle (as in the work of Impagliazzo and Rudich [65]): In each iteration, sampling $r'_{\mathcal{A}}, r'_{\mathcal{B}}$ and $\mathsf{Aug}(Q_{\mathrm{Sim}})$ can be done in polynomial space, requires access only to $Q$ which is of polynomial size and does not require access to $\Gamma$. □

## 5.6 Proof of Theorem 5.9

Equipped with Theorems 5.16, 5.18 and 5.20, we are now ready to prove Theorem 5.9.

**Proof of Theorem 5.9.** Let $(\mathcal{A}, \mathcal{B}, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of a bit-agreement protocol from a one-way function $f$ and an XiO scheme $\mathsf{xi}\mathcal{O}$ for a class of oracle-aided circuits $\mathcal{C} = \{\mathcal{C}_{s,n}\}_{s,n \in \mathbb{N}}$ relative to $\Gamma \leftarrow \mathfrak{S}_\ell$, where $\ell(s,n) = 2^{n^\epsilon} \cdot s^2$ for a constant $0 \leq \epsilon < 1$. By Theorem 5.20, there exists an oracle-aided algorithm $E$ that runs in polynomial time $T_E(s)$ such that

$$\left| \Pr\left[ \mathsf{Exp}^{\mathsf{KA}}_{\Gamma,(\mathcal{A},\mathcal{B}),E}(s) = 1 \right] - \frac{1}{2} \right| \geq \frac{7}{16},$$

where the probability is over $\Gamma \leftarrow \mathfrak{S}_\ell$, and the internal randomness of $\mathcal{A}$,$\mathcal{B}$, and $E$. By Definition 5.8, it therefore holds that either $E$ can be used to invert the one-way function $f$, or to break the security of $\mathsf{xi}\mathcal{O}$.

***E* can be used to invert the one-way function *f*.** In the first case, by Definition 5.8, it holds that

$$\Pr\left[ M^{E^{\mathsf{PSPACE}},\Gamma}(f(x)) \in f^{-1}(f(x)) \right] \geq \epsilon_{M,1}\left( \frac{16}{7} \cdot T_E(s) \right) \cdot \epsilon_{M,2}(s).$$

for infinitely many values of $s \in \mathbb{N}$, where the probability is taken over the choice of $s \leftarrow \{0,1\}^s$ and over the internal randomness of $M$. The algorithm $M$ may invoke $E$ on various security parameters (i.e., in general $M$ is not restricted to invoking $E$ only on the security parameter $s$), and we denote by $L(s)$ the maximal security parameter on which $M$ invokes $E$ (when $M$ itself is invoked on the security parameter $s$). This, viewing $M^E$ as a single oracle-aided algorithm that has access to a $\mathsf{PSPACE}$-complete oracle and to $\Gamma$, its running times $T_{M^E}(s)$ satisfies $T_{M^E}(s) \leq T_M(s) \cdot T_E(L(s))$, as $M$ may invoke $E$ at most $T_M(s)$-times, and the running time of $E$ on each invocation is at most $T_E(L(s))$. Viewing $M' \stackrel{\mathsf{def}}{=} M^{E^{\mathsf{PSPACE}}}$ as a single oracle-aided algorithm that has oracle access to $\Gamma$, this implies that $M'$ is a $q$-query algorithm where $q(s) = T_{M^E}(s)$. Theorem 5.16 then implies that either $q(s) \geq 2^{s/20}$ or $\epsilon_{M,1}\left(T_E(s) \cdot 16/7\right) \cdot \epsilon_{M,2}(s) \leq 2^{-s/2}$. We have:

- In the first case (i.e., $q(s) \geq 2^{s/20}$), noting that $L(s) \leq T_M(s)$, we obtain that

$$2^{s/20} \leq q(s) = T_{M^E}(s) \leq T_M(s) \cdot T_E(L(s)) \leq T_E(s) \cdot T_E(T_M(s)) \ .$$

The running times $T_E(s)$ of the adversary $E$ (when given access to a PSPACE-complete oracle) is some fixed polynomial in $s$, and therefore $T_M(s) \geq 2^{\gamma s}$ for some constant $\gamma > 0$.

- In the second case, i.e., $\epsilon_{M,1}\left(T_E(s) \cdot 16/7\right) \cdot \epsilon_{M,2}(s) \leq 2^{-s/2}$, since $T_E(s) < s^d$ for some constant $c$, we obtain that $\epsilon_{M,1}(s^d) \cdot \epsilon_{M,2}(s) \leq 2^{-s/2}$ for some constant $d \geq 1$.

***E* can be used to break xi$\mathcal{O}$.** In the second case we obtain from Definition 5.8 that

$$\left|\Pr\left[\mathsf{Exp}^{XiO}_{(f,\mathsf{xi}\mathcal{O}),\mathsf{xi}\mathcal{O},M^E,\mathcal{C}}(s)=1\right]-\frac{1}{2}\right| \geq \epsilon_{M,1}\left(\frac{16}{7}\cdot T_E(s)\right)\cdot\epsilon_{M,2}(s)$$

for infinitely many values of $s \in \mathbb{N}$, where $\Gamma \leftarrow \mathfrak{S}_\ell$. As in the previous case, viewing $M' \stackrel{\mathsf{def}}{=} M^{E^{\mathsf{PSPACE}}}$ as a single oracle-aided algorithm that has oracle access to $\Gamma$, implies that $M'$ is a $q$-query algorithm where $q(s) = T_{M^E}(s)$. Theorem 5.18 then implies that either $2^{s/20} \leq q(s)$ or $\epsilon_{M,1}(T_E(s) \cdot 16/7) \cdot \epsilon_{M,2}(s) \leq 2^{-s/4}$. As previously, this implies that either $T_M(s) \geq 2^{\gamma s}$ for some constant $\gamma > 0$, or $\epsilon_{M,1}(s^d) \cdot \epsilon_{M,2}(s) \leq 2^{-s/4}$ for some constant $d > 1$.

□

# 6 Compressing Obfuscation with Statistical Security

In this section we study the possibility for compressing obfuscation with perfect (information-theoretic) security. We will distinguish between approximately correct and perfectly correct compressing obfuscators and show almost tight results.

For approximately correct obfuscators, one the one hand, we show that there exists a statistically secure compressing obfuscator for the class of bounded depth circuits. On the other hand, we show that this is almost tight as any class that contains a (puncturable) PRF cannot be obfuscated with statistical secure (under complexity theoretic conjectures). See Theorems 6.4 and 6.6 for the precise parameters.

For perfectly correct obfuscators, on the one hand, we show that there exists a statistically secure compressing obfuscator for the class of bounded depth circuits, but the compression factor will be very weak (the obfuscation time is $\mathsf{poly}(2^n)$). On the other hand, we show that even for depth two circuits, better compression with better running time is implausible. See Theorems 6.2 and 6.8 for the precise parameters.

## 6.1 Negative Results

We show that it is unlikely that there is a statistically secure compressing obfuscator with good enough compression.

Our first result says that if such an obfuscator exists with strong enough compression, namely a $(2^{\epsilon n}, 2^{\epsilon n})$-compressing obfuscator with statistical security and perfect correctness, then $\overline{\mathsf{SAT}}$ (the problem of deciding whether a SAT formula is unsatisfiable) has an AM protocol in which the verifier's running time is bounded by $2^{\epsilon n}$. This is not believed to be likely for small enough values of $\epsilon > 0$, according to the best of our knowledge. Note that for this result we only need an obfuscator for depth-2 circuits. This argument relies on ideas from [70] and can be seen as an extension of an argument from [56].

**Definition 6.1.** *We denote by* $\mathsf{AM}[t,\ell]$ *the class of all languages on instances of size* $n$ *that have an* AM *protocol in which the running time of the verifier is at most* $t(n)$ *and its messages size is at most* $\ell(n)$*. The class* $\mathsf{coAM}[t,\ell]$ *is defined, analogously, to be the class that contains all the complement languages. In case that* $t = \ell$*, we will write* $\mathsf{AM}[t]$ *to denote* $\mathsf{AM}[t,t]$ *and* $\mathsf{coAM}[t]$ *to denote* $\mathsf{coAM}[t,t]$*.*

**Theorem 6.2.** *There exists a universal constant $c > 0$ such that the following holds. If there is $0 < \epsilon < 1$ and a statistically secure and perfectly correct $(2^{\epsilon n}, 2^{\epsilon n})$-compressing obfuscation for depth-2 circuits, then $\overline{\mathsf{SAT}} \in \mathsf{AM}[2^{c\epsilon n}]$.*

The conclusion in Theorem 6.2 can be stated more generally as a conjecture that is interesting on its own right. This conjecture is parametrized by an $0 < \epsilon < 1$ and it says that $\overline{\mathsf{SAT}}$ is not in $\mathsf{AM}[2^{\epsilon n}]$.

**Definition 6.3** (Conjecture)**.** *There exist $\epsilon > 0$ for which $\overline{\mathsf{SAT}} \notin \mathsf{AM}[2^{\epsilon n}]$.*

It is known that the conjecture is *false* for $\epsilon = 1/2$ by the recent result of Williams [92] who showed that $\overline{\mathsf{SAT}} \in \mathsf{AM}[\tilde{O}(2^{n/2})]$. However, for smaller values of $\epsilon$ it is still unknown. The conjecture is particularly appealing in the case that $\epsilon$ is sub-constant (some $o(1)$).

Additionally, we give evidence that a compressing obfuscator with statistical security and only *approximate* correctness cannot exist for classes of functions that contain a (puncturable) PRF. This argument relies on and extends the proof of [29].

**Theorem 6.4.** *[Restatement of Theorem 1.2, part II] There exists a universal constant $c > 0$ such that the following holds. If there is $0 < \epsilon < 1$ and a statistically secure and approximately correct $(2^{n^\epsilon}, 2^{n^\epsilon})$-compressing obfuscation for all circuits, then $\overline{\mathsf{SAT}} \in \mathsf{AM}[2^{n^\epsilon}]$.*

*Proof of Theorem 6.2.* Our proof will work by constructing a compressing (2-round) SZK protocol for all $\mathsf{NP}$ (in the analogue sense of the nontrivial AM above where the verifier's running time and message size are of slightly nontrivial size). Then, we observe that this protocol can be used to get a protocol for for the complement of $\mathsf{NP}$, thereby implying that $\overline{\mathsf{NP}}$ has a nontrivial $\mathsf{AM}$ protocol.

We define the class $\mathsf{HVSZK}[t, \ell]$ to consists of all languages for which there is an (honest-verifier) statistical zero-knowledge protocol in which the verifier runs in time at most $t$ and sends a message of size at most $\ell$. We show that compressing obfuscation with statistical security implies a nontrivial SZK protocol for all $\mathsf{NP}$.

**Claim 6.5.** *If statistically secure and perfectly correct $(t, \ell)$-compressing obfuscation $\mathcal{O}$ exists, then $\mathsf{NP} \subseteq \mathsf{HVSZK}[t, \ell]$.*

Note that when $t = 2^n$, where $n$ is the input size to the $\mathsf{NP}$ instance, it is true that $\mathsf{NP} \subseteq \mathsf{HVSZK}[t, \ell]$ since the verifier can solve the instance by itself. However, to the best of our knowledge, as long as $\ell = t \ll 2^n$ (say, $t = 2^{n^\epsilon}$ or even $t = 2^{\epsilon n}$ for small $\epsilon > 0$) it is not believed to hold. Thus, the above claim is useful only when $t \ll 2^n$.

*Proof of Claim 6.5.* We construct such a (2-round) protocol for a language $L \in \mathsf{NP}$ with associated relation $R_L$. In this protocol, the prover gets an instance $x$ and a witness $w$ and the verifier gets only the instance. Let $\Pi^s_x(w)$ be a circuit that outputs $s$ if $w \in R_L(x)$; otherwise, it outputs $\perp$. The verifier $V$ on input a statement $x \in \{0,1\}^n$ picks a random $s \leftarrow \{0,1\}^n$, generates an obfuscation $C \leftarrow \mathcal{O}(\Pi^s_x)$ and sends it to the prover. The prover $P$, on input $x$, a witness $w$, and receiving $C$ from $V$, lets $s' \leftarrow C(w)$ and sends $s'$ back to $V$. $V$ accepts if and only if $s = s'$.

The protocol is complete since if the prover has a valid witness $w$, she can evaluate the obfuscated circuit, get $s$, and send it back to the verifier. Also, perfect honest-verifier zero-knowledge holds since we can construct a simulator that simulates the whole view of the verifier. The simulator samples a random tape for the verifier, which includes $s$ and just outputs it.

To show soundness, consider some cheating prover $P^*$ that convinces $V$ with inverse polynomial probability $1/p(|x|)$ for infinitely many $x \notin L$. Consider some $x \notin L$. Note that $\Pi^s_x$ is functionally equivalent to the "dummy" circuit $\Pi^\perp$ that always outputs $\perp$. Thus, by the indistinguishability property of $\mathcal{O}$, $C$ is indistinguishable from $C' = \mathcal{O}(\Pi^\perp)$. It follows that in a modified experiment where $V$ sends $C'$ instead of $C$, $P^*$ also convinces $V$ with inverse polynomial probability $1/p'(|x|)$

for infinitely many $x \notin L$. However, in this experiment $P^*$'s view is independent of $s$ and it can thus only guess $s$ with probability $2^{-|s|}$, which is a contradiction. □

Next, by applying the transformation of Okamoto [86], we can transform the above HVSZK protocol into a HVSZK protocol for $\mathsf{coNP}$. The transformation is done in two steps. First, the HVSZK protocol is turned into a public-coin HVSZK protocol, where the verifier's messages are just its coin flips. Applying this transformation, we get a verifier whose running time is a fixed polynomial in the running time of the simulator of the original protocol. Second, we transform the latter HVSZK public-coin protocol into an HVSZK protocol for the complement language (i.e., $\mathsf{coNP}$). This step also blows up the complexity of the verifier by a fixed polynomial in the running time of the simulator of the protocol we started with (the public-coin one).

Overall, the overhead in the transformation above is some fixed polynomial in the running time of the simulator of the original protocol. Let $c \in \mathbb{N}$ be the exponent of this polynomial. Thus, since the simulator runs in time at most $t(n) = 2^{\epsilon n}$, then the complexity of the verifier in the new HVSZK protocol will be a fixed polynomial in $t(n)$, namely $2^{c\epsilon n}$. This completes the proof since:

$$\overline{\mathsf{SAT}} \in \mathsf{coNP} \subseteq \mathsf{SZK}[2^{c\epsilon n}, 2^{c\epsilon n}] \subseteq \mathsf{AM}[2^{c\epsilon n}, 2^{c\epsilon n}].$$

□

*Proof of Theorem 6.4.* We will largely follow the argument in [29] who showed an analogous result for iO. Let us sketch their argument. Based on puncturable PRFs and an approximately correct statistically secure iO, the construct a distribution over pairs of circuits (that will be later indexed by $\mathsf{SAT}$ formulas) such that the circuits differ only on one point and yet the obfuscator will produce distributions that are statistically far.

Let $k$ be a key for a puncturable PRF family $F$, let $x_0$ be a random point in the domain, and let $k\{x_0\}$ be the punctured key $k$ at the point $x_0$. They consider the function $f_{k\{x_0\},y}$ that, on input $x$ outputs $F_k(x)$ if $x \neq x_0$, and outputs $y$ if $x = x_0$. On the one hand, by definition, $f_{k\{x_0\},y}$ for a random $y$ and $f_{k\{x_0\},y_0}$ for $y = F_k(x_0)$, are functionally equivalent at any point except maybe at $x_0$. On the other hand, by the security of the puncturable PRF, when $k$, $x_0$, and $y$ are chosen at random the distributions $\mathsf{i}\mathcal{O}(f_{k\{x_0\},y})$ and $\mathsf{i}\mathcal{O}(f_{k\{x_0\},y_0})$, are statistically *far*.

They use this idea to distinguish between (uniquely) satisfiable and unsatisfiable formulas. The idea is to hardwire in $f$ the formula $\psi$ and instead of checking whether $x = x_0$, we check $\psi(x) = 1$ and if so output the hardwired point $y$. To make the argument work, they need $\psi(x) = 1$ to hold (if it holds at some point) at a random point, so they hardwire a randomly "shifted" version of the formula. Now, the above argument can be repeated and the result is that $\mathsf{USAT} \in \mathsf{BPP}^{\mathsf{GapSD}}$, where $\mathsf{USAT}$ is the problem of deciding whether a SAT formula is uniquely satisfiable and where $\mathsf{GapSD}$ is the $\mathsf{SZK}$ complete problem [89] that requires to distinguish between efficient samplers for statistically close distributions from statistically far distributions. They then apply an argument of Mahmoody and Xiao [82] that says that if $\mathsf{USAT} \in \mathsf{BPP}^{\mathsf{GapSD}}$, then $\mathsf{SAT} \in \mathsf{AM} \cap \mathsf{coAM}$.

We will repeat the above argument with a $(2^{n^\epsilon}, 2^{n^\epsilon})$-compressing obfuscator as assumed in the statement. The only change we need to make is to modify the circuit $f_{k\{x_0\},y}$ to accept inputs of size $n' = \log^{1/\epsilon} n$ so that an obfuscation is of size at most polynomial in $n$. Denote the $\mathsf{USAT}$ problem on formulas with $n'$ variables by $\mathsf{USAT}[n']$. The above argument shows that $\mathsf{USAT}[n'] \in \mathsf{BPP}^{\mathsf{GapSD}}$. By the result of [82], this implies that $\mathsf{SAT}[n'] \in \mathsf{AM} \cap \mathsf{coAM}$, or in other words that $\overline{\mathsf{SAT}}[n'] \in \mathsf{AM}$. The result in the statement now follows by scaling the parameters and applying the result with a formula with $n'' = 2^{n^\epsilon}$. □

## 6.2 Positive Results

We show that for small classes of circuits there is a compressing obfuscation with perfect security. We start with the constructions that give approximate correctness.

**Theorem 6.6.** *[Restatement of Theorem 1.2, part I] There exist constants $0 < \alpha < 1$ and $0 < \beta < 1$ such that there exists a $(1 - s/2^{n^\beta})$-approximately correct $(2^{n^\alpha}, 2^{n^\alpha})$-compressing obfuscator with perfect security for the class of polynomial-size constant-depth $n$-input Boolean circuits.*

**Theorem 6.7.** *There exists a polynomial $p(\cdot)$ and a constant $\alpha > 0$ such that there exists a $(1 - 1/p(n))$-approximately correct $(2^{(1-\alpha)n}, 2^{(1-\alpha)n})$-compressing obfuscator with perfect security for the class of monotone $n$-input Boolean functions.*

We show that the class of bounded-depth circuits above can also be obfuscated with perfect correctness, while still resulting with a compressing obfuscator. However, the resulting compression is very weak (in particular, such compression, even for compressing obfuscation for all circuits is not known to imply full-fledged obfuscation).

**Theorem 6.8.** *[Restatement of Theorem 1.3] There exists a perfectly correct $(\mathsf{poly}(2^n), 2^{n-n/O(\log s)^{d-1}})$-obfuscator with perfect security for the class of size $s$ depth $d$, $n$-input Boolean circuits.*

All of the obfuscators above treat their input circuit as a black box and run a classical *learning* or *compression* algorithm on it. We introduce these tasks next.

**Preliminaries on PAC learning.** We begin by introducing the concept of PAC learning. The Probably Approximately Correct (PAC) learning model, introduced by Valiant [91], is one of the most central definitions in the learning community and in computer science in general. We focus on PAC learning over the uniform distribution with membership queries. In this setting the learner may query the oracle at any point $x$ and get back the value of the oracle at that point.

**Definition 6.9** (PAC learning over the uniform distribution with membership queries)**.** *Let $\mathcal{F}$ be a class of Boolean functions over $n$ inputs. The class $\mathcal{F}$ is $(\epsilon, \delta)$-PAC learnable if there exists an algorithm $\mathcal{A}$ that gets as input two parameters $\epsilon, \delta > 0$, has membership query access to a function $f \in \mathcal{F}$, and outputs with probability $1 - \delta$ (over its internal randomness) a circuit $C$ that agrees with $f$ on all but an $\epsilon$-fraction of the inputs. That is,*

$$\Pr_{\mathcal{A}}\left[C \leftarrow \mathcal{A}^f(\epsilon, \delta); \Pr_{x \leftarrow \{0,1\}^n}[C(x) \neq f(x)] \leq \epsilon\right] \geq 1 - \delta.$$

*The running time of $A$ is measures as a function of $n, 1/\epsilon, 1/\delta$, and the circuit size of $f$.*

There has been a tremendous amount of work on obtaining efficient algorithms for PAC learning various classes of functions (see [62] for a survey). It is known that no $\mathsf{poly}(n)$-time algorithm can learn arbitrary Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ to accuracy non-negligibly better than $1/2$, but many positive results are known for restricted classes of functions. We fix $\delta = 2/3$, and note that this choice is somewhat arbitrary and enough for all of our applications. We thus say that a class is $\epsilon$-PAC learnable if it is $(\epsilon, 2/3)$-PAC learnable.

One well known example is the quasi-polynomial time algorithm of Linial, Mansour, and Nisan [78] for the class of functions computed by $\mathsf{AC}^0$ circuits (constant depth circuits with AND, OR, and NOT gates of unbounded fan-in and fan-out).

**Theorem 6.10** (Learning bounded-depth circuits [78])**.** *The class of size-$s$ depth-$d$ circuits is $\epsilon$-PAC learnable within $n^{O(\log^{d-1}(s/\epsilon))}$ queries.*[9]

Another notable example that is relevant for us is the algorithm of Bshouty and Tamon [34] for learning arbitrary monotone functions.

[9] In Theorems 6.10 and 6.11 it is enough that the labels are for uniformly random inputs (i.e., random examples).

**Theorem 6.11** (Learning monotone functions [34])**.** *The class of monotone functions is $\epsilon$-PAC learnable within $n^{O(\sqrt{n}/\epsilon)}$ queries.*

A more recent result of Carmosino et al. [35] showed a (quasi-polynomial-time) learner for $\mathsf{AC}^0[p]$, the class of Boolean constant depth circuits with unbounded fan-in and fan-out with AND, OR, NOT, and MOD-$p$ gates.[10]

**Theorem 6.12** (Learning bounded-depth circuits with $\mathsf{mod}$ gates [35])**.** *For every prime $p > 1$, the class of $\mathsf{AC}^0[p]$ circuits of size $s$ is $\epsilon$-PAC learnable within $2^{\mathsf{poly}\log(ns/\epsilon)}$ queries.*

We are now ready to show that the above learning procedures imply the claimed obfuscators.

*Proof of Theorem 6.6.* Given an $n$-input circuit $C$ of size $s$ and depth $d$, we obfuscate it by running the learning algorithm from Theorem 6.10, simulating each oracle query with input $x$ by executing $C$ on $x$ and returning the reply.

It is guaranteed that the resulting circuit is of size $n^{O(\log^{d-1}(s/\epsilon))}$ and it approximates the original circuit on all but $\epsilon$ fraction of the inputs. Since the dependence on $1/\epsilon$ is logarithmic in the exponent we can choose it to be $\epsilon = \frac{s}{2^{\,2d-2\sqrt[2]{n}}}$. This bounds the running time of the learner (and thus its output size) by

$$n^{O(\log^{d-1}(s/\epsilon))} \leq 2^{\log n \cdot O(\log^{d-1}(2^{\,2d-2\sqrt[2]{n}}))} = 2^{O(\sqrt{n}\cdot\log n)}.$$

Since our obfuscator treats its input circuit as a black-box, the resulting obfuscation can be perfectly simulated with only oracle access to the circuit. □

*Proof of Theorem 6.7.* Given an $n$-input circuit $C$ that computes a monotone function we obfuscate it by running the learning algorithm from Theorem 6.11, simulating each oracle query with input $x$ by executing $C$ on $x$ and returning the reply.

It is guaranteed that the resulting circuit is of size $n^{O(\sqrt{n}/\epsilon)}$ and it approximates the original circuit on all but $\epsilon$ fraction of the inputs. We set $\epsilon = 1/n^{0.499}$ and get that the running time of the obfuscator and size of the resulting circuit are bounded by $2^{0.9999n}$. As before, since our obfuscator treats the input circuit as a black-box, the resulting obfuscation is perfectly secure. □

**Tightness of the approach.** The approach of constructing obfuscators via learning algorithms is inherently limited. As observed by Valiant [91], any class that contains a pseudorandom function cannot be learned with nontrivial savings. Moreover, this approach, as shown above, gives the very strong notion of perfect security, which does not exist for all functions (even the computational version, known as virtual black-box, does not exist for circuits that contain a PRF [15]). Thus, to get an obfuscator (that satisfies only indistinguishability obfuscation) for a larger class of functions, one has to use the fact that the obfuscator has access to a circuit rather than treating it as a black-box.

**Preliminaries on circuit compression.** In the problem of circuit compression, studied by Chen et al. [37], one is given the truth table of a Boolean function $f$ computable by some *unknown* circuit from a known class of circuits, and the goal is to find in time $\mathsf{poly}(2^n)$ a circuit $C$ (not necessarily from the aforementioned family) computing $f$ so that the size of $C$ is less than the trivial circuit size $\approx 2^n$. For general functions this is impossible as a counting argument shows that there are functions that require this size, so the focus is on restricted classes.

[10] Recently, Carmosino et al. [36] generalized their result to get an implication from "tolerant" natural proofs to agnostic learning [66]. In agnostic learning, is the same as in PAC learning except that the learner is only guaranteed that $f$ is close to the concept class $\mathcal{C}$ (rather than assuming it belongs to it).

**Definition 6.13** ($\mathcal{C}$-compression)**.** *Given the truth table of an $n$-variate Boolean function $f \in \mathcal{C}$, find a Boolean circuit of size $< 2^n/n$ that is functionally equivalent to $f$.*

As mentioned in [37], compression of Boolean functions is related to the setting of exact learning with membership and equivalence queries [7]. In this learning setting, the size of the hypothesis produced by the learning algorithm is upper-bounded by the running time of the algorithm. In the circuit compression setting, the hypothesis (compressed image) size and the running time of the learning (compression) algorithm are decoupled: we allow more running time, but ask for a small-size compression. This may enable improvements in the class of circuits that we can handle. Concretely, exact learning is strictly stronger as any result in exact learning yields a compression algorithm for the corresponding class of functions, but the opposite direction is not known.

We notice that in general good enough compression implies compressing obfuscation where the output size is nontrivial but the running time can be large enough to read the truth table of the function (i.e., as in XiO). However, the other direction is not known since in the obfuscation setting one is given a witness (i.e., a circuit rather than the truth table). The most relevant circuit compression result that is relevant for us is stated next.

**Theorem 6.14** ([37])**.** *If a Boolean $n$-variate function is computed by an $\mathsf{AC}^0$ circuit of size $s$ and depth $d$, then it is compressible to a circuit of size at most $2^{n-n/O(\log s)^{d-1}}$.*

As in the case of learning algorithms, the above compression algorithm directly implies a perfectly correct compressing obfuscator satisfying perfect security. We will avoid repetition and skip the proof of Theorem 6.8 (which follows directly from Theorem 6.14).

Note that, as in the case of learning, it is impossible to compress a class of circuits that contains a PRF. For this, consider a PRF with key size $n^2$ and input size $n$ which is exponentially secure (namely, secure for adversaries running in time $2^{\Omega(n^2)}$).[11] In this case, the PRF-or-Random adversary is allowed to query the oracle at all $2^n$ inputs and yet it still cannot distinguish PRF from random. The impossibility of compression for such a family of circuits now follows from the fact that random functions cannot be compressed.

## Acknowledgments

We thank Zvika Brakerski for discussions about the possibility of SXiO and XiO with statistical security.

## References

[1] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology - CRYPTO*, pages 657–677, 2015.

[2] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *Advances in Cryptology - CRYPTO*, pages 491–520, 2016.

[3] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Robust transforming combiners from indistinguishability obfuscation to functional encryption. In *Advances in Cryptology - EUROCRYPT*, pages 91–121, 2017.

[11] The argument works even with sub-exponential security by increasing the size of the key.

[4] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology - CRYPTO*, pages 308–326, 2015.

[5] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *Advances in Cryptology - EUROCRYPT*, pages 152–181, 2017.

[6] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding barrington's theorem. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 646–658, 2014.

[7] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.

[8] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In *44th International Colloquium on Automata, Languages, and Programming, ICALP*, pages 38:1–38:16, 2017.

[9] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In *Theory of Cryptography - TCC*, pages 528–556, 2015.

[10] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM Journal on Computing*, 45(6):2117–2176, 2016.

[11] Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Theory of Cryptography Conference*, 2016.

[12] C.A. Asmuth and G.R. Blakley. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Computers & Mathematics with Applications*, 7(6):447 – 450, 1981.

[13] Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In *Advances in Cryptology - EUROCRYPT*, pages 649–679, 2018.

[14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In *Advances in Cryptology - EUROCRYPT*, pages 221–238, 2014.

[15] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[16] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In *Advances in Cryptology - CRYPTO*, pages 696–723, 2017.

[17] Nir Bitansky, Huijia Lin, and Omer Paneth. On removing graded encodings from functional encryption. In *Advances in Cryptology - EUROCRYPT*, pages 3–29, 2017.

[18] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From Cryptomania to Obfustopia through secret-key functional encryption. In *Theory of Cryptography - TCC*, pages 391–418, 2016.

[19] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography - TCC*, pages 401–427, 2015.

[20] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography - TCC*, pages 474–502, 2016.

[21] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pages 171–190, 2015.

[22] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation: From approximate to exact. In *Theory of Cryptography - TCC*, pages 67–95, 2016.

[23] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. In *Advances in Cryptology - EUROCRYPT*, pages 592–606, 2017.

[24] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT*, pages 533–556, 2014.

[25] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[26] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT*, pages 280–300, 2013.

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

[28] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography - PKC*, pages 501–519, 2014.

[29] Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. On statistically secure obfuscation with approximate correctness. In *Advances in Cryptology - CRYPTO*, pages 551–578, 2016.

[30] Zvika Brakerski, Aayush Jain, Ilan Komargodski, Alain Passelègue, and Daniel Wichs. Non-trivial witness encryption and null-io from standard assumptions. *IACR Cryptology ePrint Archive*, 2017:874, 2017.

[31] Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In *Theory of Cryptography - TCC*, pages 559–578, 2011.

[32] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *J. Cryptology*, 31(2):434–520, 2018.

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

[34] Nader H. Bshouty and Christino Tamon. On the fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996.

[35] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *31st Conference on Computational Complexity, CCC*, pages 10:1–10:24, 2016.

[36] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic learning from tolerant natural proofs. In *Approximation, Randomization, and Combinatorial Optimization, APPROX/RANDOM*, pages 35:1–35:19, 2017.

[37] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

[38] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In *Advances in Cryptology - EUROCRYPT*, pages 278–307, 2017.

[39] 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*, pages 3–12, 2015.

[40] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology - CRYPTO*, pages 247–266, 2015.

[41] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Advances in Cryptology - CRYPTO*, pages 476–493, 2013.

[42] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. New multilinear maps over the integers. In *Advances in Cryptology - CRYPTO*, pages 267–286, 2015.

[43] Whitfield Diffie and Martin E. Hellman. Multiuser cryptographic techniques. In *American Federation of Information Processing Societies*, pages 109–112, 1976.

[44] Marc Fischlin, Amir Herzberg, Hod Bin Noon, and Haya Shulman. Obfuscation combiners. In *Advances in Cryptology - CRYPTO*, pages 521–550, 2016.

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

[46] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Symposium on Theory of Computing Conference, STOC*, pages 467–476, 2013.

[47] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ameer Mohammed. Limits on the power of garbling techniques for public-key encryption. In *Advances in Cryptology - CRYPTO*, pages 335–364, 2018.

[48] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In *Advances in Cryptology - CRYPTO*, pages 661–695, 2017.

[49] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. When does functional encryption imply obfuscation? In *Theory of Cryptography - TCC*, pages 82–115, 2017.

[50] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *Theory of Cryptography - TCC*, pages 498–527, 2015.

[51] Craig Gentry, Allison Bishop Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS*, pages 151–170, 2015.

[52] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*, chapter 4.10.3.1. Cambridge University Press, 2001.

[53] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[54] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Advances in Cryptology - EUROCRYPT*, pages 578–602, 2014.

[55] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC*, pages 555–564, 2013.

[56] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *Theory of Cryptography - TCC*, pages 194–213, 2007.

[57] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology - CRYPTO*, pages 162–179, 2012.

[58] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory, 2013. `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/index.html`. Accessed May 31, 2018.

[59] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the 32nd annual ACM symposium on Theory of computing, STOC*, pages 181–190. ACM, 2000.

[60] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology - EUROCRYPT*, pages 96–113, 2005.

[61] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[62] Lisa Hellerstein and Rocco A. Servedio. On PAC learning algorithms for rich boolean function classes. *Theor. Comput. Sci.*, 384(1):66–76, 2007.

[63] Amir Herzberg. On tolerant cryptographic constructions. In *Topics in Cryptology - CT-RSA*, pages 172–190, 2005.

[64] Amir Herzberg. Folklore, practice and theory of robust combiners. *Journal of Computer Security*, 17(2):159–189, 2009.

[65] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st annual ACM symposium on Theory of computing, STOC*, pages 44–61. ACM, 1989.

[66] Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[67] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.

[68] Sam Kim and David J. Wu. Multi-theorem preprocessing nizks from lattices. In *Advances in Cryptology - CRYPTO*, 2018.

[69] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In *Advances in Cryptology - EUROCRYPT*, pages 603–648, 2018.

[70] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 374–383, 2014.

[71] Ilan Komargodski and Gil Segev. From Minicrypt to Obfustopia via private-key functional encryption. In *Advances in Cryptology - EUROCRYPT*, pages 122–151, 2017.

[72] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[73] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT*, pages 28–57, 2016.

[74] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *Advances in Cryptology - CRYPTO*, pages 599–629, 2017.

[75] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public-Key Cryptography - PKC*, pages 447–462, 2016.

[76] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Output-compressing randomized encodings and applications. In *Theory of Cryptography - TCC*, pages 96–124, 2016.

[77] 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*, pages 11–20, 2016.

[78] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science, FOCS*, pages 574–579, 1989.

[79] Qipeng Liu and Mark Zhandry. Decomposable obfuscation: A framework for building applications of obfuscation from polynomial hardness. In *Theory of Cryptography - TCC*, pages 138–169, 2017.

[80] Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *Theory of Cryptography - TCC*, pages 119–137, 2017.

[81] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In *Theory of Cryptography - TCC*, pages 49–66, 2016.

[82] Mohammad Mahmoody and David Xiao. On the power of randomized reductions and the checkability of SAT. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC*, pages 64–75. IEEE Computer Society, 2010.

[83] Silvio Micali, Chris Peikert, Madhu Sudan, and David A Wilson. Optimal error correction against computationally bounded noise. In *Theory of Cryptography - TCC*, pages 1–16. Springer, 2005.

[84] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In *Advances in Cryptology - CRYPTO*, pages 629–658, 2016.

[85] Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.

[86] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000.

[87] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010:556, 2010.

[88] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In *Advances in Cryptology - CRYPTO*, pages 500–517, 2014.

[89] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[90] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Symposium on Theory of Computing, STOC*, pages 475–484, 2014.

[91] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[92] Richard Ryan Williams. Strong ETH breaks with merlin and arthur: Short non-interactive proofs of batch evaluation. In *31st Conference on Computational Complexity, CCC*, pages 2:1–2:17, 2016.

[93] Joe Zimmerman. How to obfuscate programs directly. In *Advances in Cryptology - EUROCRYPT*, pages 439–467, 2015.