The Pseudorandom Oracle Model and Ideal Obfuscation

Aayush Jain[1] Huijia Lin[2] 罗辑 (Ji Luo)[2] Daniel Wichs[3,4]

[1] Carnegie Mellon University
aayushja@andrew.cmu.edu

[2] University of Washington
{rachel,luoji}@cs.washington.edu

[3] Northeastern University
wichs@ccs.neu.edu

[4] NTT Research

10 August 2023

Abstract

We introduce a new idealized model of hash functions, which we refer to as the *pseudorandom oracle* (Pr$\mathcal{O}$) model. Intuitively, it allows us to model cryptosystems that use the code of an ideal hash function in a non-black-box way. Formally, we model hash functions via a combination of a pseudorandom function (PRF) family and an ideal oracle. A user can initialize the hash function by choosing a PRF key k and mapping it to a public handle h using the oracle. Given the handle h and some input x, the oracle can also be called to evaluate the PRF at x with the corresponding key k. A user who chooses the PRF key k therefore has a complete description of the hash function and can use its code in non-black-box constructions, while an adversary, who just gets the handle h, only has black-box access to the hash function via the oracle.

As our main result, we show how to construct ideal obfuscation in the Pr$\mathcal{O}$ model, starting from functional encryption (FE), which in turn can be based on well-studied polynomial hardness assumptions. In contrast, we know that ideal obfuscation cannot be instantiated in the basic random oracle model under any assumptions. We believe our result provides heuristic justification for the following: (1) most natural security goals implied by ideal obfuscation can be achieved in the real world; (2) obfuscation can be constructed from FE at polynomial security loss.

We also discuss how to interpret our result in the Pr$\mathcal{O}$ model as a construction of ideal obfuscation using simple hardware tokens or as a way to bootstrap ideal obfuscation for PRFs to that for all functions.

This is the full version (a major revision) of [JLLW23] (in the proceedings of Crypto 2023, published by Springer, © IACR 2023, DOI 10.1007/978-3-031-38551-3_8).

Contents

1 Introduction **1**

1.1 Basics of the Pseudorandom Oracle (Pr$\mathcal{O}$) Model 2

1.2 Interpreting Our Result of Ideal Obfuscation 3

1.3 Further Discussion on the Pr$\mathcal{O}$ Model 5

1.4 Related Works 7

2 Technical Overview **8**

3 Preliminaries **13**

4 The Pseudorandom Oracle (Pr$\mathcal{O}$) Model **16**

5 Ideal Obfuscation **16**

6 Construction of Ideal Obfuscation in the Pr$\mathcal{O}$ Model **17**

7 Security Proof of Ideal Obfuscation in the Pr$\mathcal{O}$ Model **21**

7.1 Simulator 24

7.2 Hybrids over Levels 24

7.3 Hybrids over Blocks at Each Level 28

7.4 Choice of Parameters 30

References **31**

1 Introduction

Hash Functions and Random Oracles. Hash functions are one of the most important cryptographic primitives and are ubiquitous in both theoretical and practical cryptosystem designs. The basic security property of a hash function is collision resistance, which is already sufficient for many applications. However, there is a widespread belief that good hash functions can satisfy a much wider range of security properties beyond collision resistance. This belief is captured in the *random oracle model* (ROM) [BR93], where we model a hash function as a truly random public function and give the honest users as well as the adversary oracle access to this function. The random oracle is an ideal functionality relative to which we can construct cryptosystems and formally prove their security. We then take a heuristic leap of faith that such cryptosystems remain secure even when we replace the random oracle by a real, well-designed hash function (like SHA-3). While the second step is heuristic and has no formal justification, it captures the intuition that an adversary cannot do anything meaningful with a well-designed hash function other than treating it as a random oracle. The random oracle heuristic is immensely popular and successful. Almost all cryptosystems used in practice, from TLS to Bitcoin, rely on it to justify their security. On the theory side, there are contrived examples where the random oracle heuristic fails — specially designed cryptosystems that are provably secure in the random oracle model, but are insecure when instantiated with any real hash function [Bar01,GK03]. Nevertheless, outside of such specially crafted counterexamples, the random oracle heuristic gives extremely strong evidence of security in real life, and there is no known example of it ever leading to a security flaw in a natural real-world cryptosystem.

Indistinguishability or Ideal Obfuscation. A scenario analogous to the one above also plays out in the upper reaches of cryptography when it comes to obfuscation [BGI+01]. We have a standard-model definition of obfuscation security called *indistinguishability obfuscation* ($i\mathcal{O}$) [BGI+01], and as of recently, we even have instantiations under well-studied assumptions [JLS21,JLS22]. While $i\mathcal{O}$ suffices for some applications, it does not suffice for many others, or results in exceedingly complex and cumbersome constructions. Similarly to hash functions, we believe obfuscators are capable of satisfying a much wider range of security properties beyond $i\mathcal{O}$. Similarly to the random oracle model for hash functions, we can define an *ideal obfuscation model*, where we model obfuscation as an ideal functionality that only gives the adversary black-box access to the obfuscated programs.[1] Analogously to the ROM, we can design cryptosystems and prove their security in the ideal obfuscation model, which is extraordinarily powerful and allows for very simple constructions. We can then make a heuristic leap of faith that such cryptosystems remain secure when we replace the ideal obfuscator by $i\mathcal{O}$. Also analogously to the random oracle model, one can come up with contrived counterexamples (e.g., [BGI+01]) where this heuristic fails, but the intuition is that it should be secure in almost all natural use cases that come up in real life.

[1] Ideal obfuscation is similar to virtual black-box (VBB) obfuscation [BGI+01], except that we consider it to be an idealized model rather than a security definition. In contrast, VBB was originally intended as a security definition, with some artificial choices (restricting adversaries to only 1-bit output) to rule out obvious counterexamples. Nevertheless, the main result of [BGI+01] shows that even with these restrictions, VBB security is unachievable in its full generality in the plain model.

Our Work: Ideal Obfuscation from Ideal Hash Functions. Summarizing the above discussion, we have the analogy that collision-resistant hash functions are to random oracles as $i\mathcal{O}$ is to ideal obfuscation. Modeling hash functions as random oracles is well-established, yet there was little study on the idealization of obfuscators. We wonder:

Can we justify the ideal obfuscation model, e.g.,
by constructing ideal obfuscation from ideal hash functions?

As a starting point, we might try constructing ideal obfuscation in the random oracle model under appropriate additional standard-model assumptions. Such a result would formalize that the ideal obfuscation heuristic is just a special case of the ROM heuristic. Unfortunately, the work of [CKP15] rules out the above attempt by showing that it is impossible to construct ideal obfuscation in the ROM.

Nevertheless, in this work, we re-examine the question of constructing ideal obfuscation from ideal hash functions, and show that it is indeed possible! To get around the previous negative result, we need to tweak our modeling of ideal hash functions. Instead of the random oracle model, we introduce a new and more flexible idealized model of hash functions that we call the *pseudorandom oracle* (Pr$\mathcal{O}$) *model*. We argue that the Pr$\mathcal{O}$ model captures the same intuition as the usual ROM, but provides more technical flexibility. As our main result, we show how to construct ideal obfuscation in the Pr$\mathcal{O}$ model. Our construction assumes (single-key, sublinearly succinct) functional encryption (FE), a strong yet standard-model primitive. We believe that this result formalizes the following intuition:

Heuristically assuming that we have ideal obfuscation
is not worse or "crazier" than
heuristically assuming that we have ideal hash functions.

As such, confidence in the latter supports confidence in the former. Furthermore, our construction of ideal obfuscation from FE in the Pr$\mathcal{O}$ model only incurs a polynomial security loss. Combined with the fact that FE can be based on well-studied polynomial assumptions [JLS21,JLS22], we obtain a heuristic obfuscator based on polynomial hardness. In contrast, constructions of $i\mathcal{O}$ from FE in the standard model incur an exponential security loss.

1.1 Basics of the Pseudorandom Oracle (Pr$\mathcal{O}$) Model

Just like the ROM, the Pr$\mathcal{O}$ model is defined in terms of a formally specified ideal functionality that all parties (honest users as well as the adversary) have access to. The ideal functionality for Pr$\mathcal{O}$ is specified relative to some (standard-model) pseudorandom function (PRF) family H_k and has two interfaces. The first interface *initializes* a hash function when provided with a PRF key k as input — the ideal functionality maps the PRF key k to a random *handle* h and outputs it. The second interface *evaluates* the hash function when provided with a handle h and an input x — the ideal functionality finds the PRF key k corresponding to h and outputs $H_k(x)$.

A Pr$\mathcal{O}$ can be used as a basic RO. Consider an honest user who chooses a random PRF key k, uses the Pr$\mathcal{O}$ to get the corresponding handle h, and then discards k and publishes h. In that case, the adversary essentially just gets oracle access to the hash function H_k by querying the oracle with the handle h. By the pseudorandomness

of H_k, this is indistinguishable from a truly random oracle.

The Pr$\mathcal{O}$ model also provides additional flexibility in allowing the honest user who chose k to use the code of the hash function H_k in a non-black-box way (e.g., inside fully homomorphic encryption, functional encryption, or garbled circuits). In other words, the Pr$\mathcal{O}$ allows different users of the cryptosystem to use different descriptions of the same hash function. The first description is given via the key k, which specifies the full code of the hash function H_k and allows for evaluating it in a non-black-box way without making any oracle calls. The second description is given via the handle h, which only provides black-box access to the hash function H_k via oracle queries. The first description is useful for functionality, but provides no security guarantee — since we only assume PRF security for H_k, if the adversary ever sees the PRF key k, all security is lost. The second description is useful for security, but provides no functionality advantage over the basic ROM. The power of the Pr$\mathcal{O}$ comes from the fact that it simultaneously gives us both descriptions for the same hash function and allows us to model different levels of access to the same hash function for different users.

However, the Pr$\mathcal{O}$ model is very conservative about what kind of security guarantees it provides, and proving security in the Pr$\mathcal{O}$ is generally very subtle and requires extreme care. In particular, if the adversary ever receives any information about k via the non-black-box use of H_k, then all security guarantees are lost! Our analysis can only make use of Pr$\mathcal{O}$ security in hybrid games where all information about the key k is removed from the view of the adversary. This implies that although our overall cryptosystem may rely on non-black-box use of the hash function, for the Pr$\mathcal{O}$ model to be helpful in security proofs, such use must be indistinguishable from black-box use in the eyes of the adversary.

Using the Pr$\mathcal{O}$. Looking ahead, it is illustrative to examine the role of the Pr$\mathcal{O}$ in our construction of ideal obfuscation. For obfuscation, we have two users with different roles — the *obfuscator* who creates the obfuscated program, and the *evaluator* who gets the obfuscated program and evaluates it on various inputs. The obfuscator will choose several PRF keys k_i and the obfuscated program will encrypt the keys k_i into some functional encryption (FE) ciphertext. The evaluator will get the FE ciphertext as well as the corresponding handles h_i, which will be sufficient for evaluating the obfuscated program on any input. In the security analysis, the adversary plays the role of the evaluator. Although it does not get keys k_i directly, it gets an FE ciphertext containing them. To argue security, we will need a careful sequence of hybrids in each of which we replace the Pr$\mathcal{O}$ outputs for some handle h_i by random values after removing the corresponding PRF key k_i from the FE ciphertext.

1.2 Interpreting Our Result of Ideal Obfuscation

Ideal obfuscation cannot be realized by any standard-model obfuscation scheme, similar to the fact that a random oracle cannot be realized by any standard-model hash function. However, also similar to random oracles, ideal obfuscation provides a formal model in which we can design and analyze cryptosystems. We can instantiate them using a real-world obfuscator based on the intuition that a good obfuscator is sufficient to achieve the security required by the cryptosystem from ideal obfuscation in most reasonable scenarios. This heuristic is powerful.

1. It allows us to reach security goals outside the current scope of standard-model proofs. The literature already contains an impressive list of such examples: virtual gray-box obfuscation [BC10], extractable witness encryption, ABE for RAM [GKP+13], input-hiding obfuscation for evasive functions [BBC+14], public-coin differing-input obfuscation, obfuscation for input-unbounded Turing machines [IPS15], doubly efficient PIR [BIPW17], FHE for RAM [HHWW19], OT over binary erasure channels [AIK+21], wiretap-channel coding [IKLS22], refuting the dream XOR lemma [BIK+22], etc. This list is sure to grow.[2]

2. It enables (conceptually) simple constructions. Consider for instance the task of building FE. Using ideal obfuscation, we can simply set the secret key for a function f to be an obfuscated program that decrypts ciphertexts of a (CCA-secure) public-key encryption and then computes the function f on the decryption result. In contrast, $i\mathcal{O}$ applications typically involve more sophisticated techniques (such as puncturing [SW14]) to overcome the weak security of $i\mathcal{O}$, producing cumbersome constructions with complex proofs of security. The ideal obfuscation heuristic gives strong evidence that such complication is actually unnecessary.

3. For many real-life security goals such as protecting software patches, creating crippleware with parts of the functionality redacted, obfuscating machine learning models, etc., $i\mathcal{O}$ security is insufficient. In these specific natural contexts, virtual black-box security is plausible (the impossibility of [BGI+01] does not apply) and can be heuristically instantiated.

When it comes to which concrete obfuscator to use when instantiating the ideal obfuscation heuristic, in the literature, the standard-model $i\mathcal{O}$ construction is typically used. Our construction of ideal obfuscation in the Pr$\mathcal{O}$ model, when instantiated with a well-designed hash function, provides another option. An advantage of our obfuscator is that it can be based on polynomial hardness assumptions, as opposed to requiring subexponential hardness.

Comparison with Best-Possible Obfuscation. Goldwasser and Rothblum showed that $i\mathcal{O}$ is a "best-possible obfuscator" [GR07], in the sense that if some obfuscator is capable of hiding some information of the program, then so does $i\mathcal{O}$. Intuitively, it says that *if* a program can be obfuscated with stronger security, then $i\mathcal{O}$ is a good candidate achieving the stronger security. However, it does not address the question when the premise is true. Our result that ideal obfuscation is feasible in the Pr$\mathcal{O}$ model gives supporting evidence that *natural* programs can be obfuscated with stronger security, under the heuristic that Pr$\mathcal{O}$ can be instantiated using real-world hash functions in *natural* scenarios. It suggests that obfuscation with stronger security is possible in most natural applications.

Alternative Interpretations: Hardware Tokens and Bootstrapping. Our result can also be interpreted as constructing ideal obfuscation using hardware tokens. The obfuscator chooses the PRF key k and releases a hardware token that implements the PRF H_k (acting as the handle in the Pr$\mathcal{O}$ model to provide black-box access to H_k)

[2] We note that after the initial pre-print and prior to the publication of this work, DE-PIR, FHE for RAM [LMW23], and ABE for TM/RAM [ACFQ22,JLL23] are achieved in the standard model.

and the obfuscated program containing encrypted k. There are several prior works showing how to construct obfuscation using hardware tokens [DMMN11,BCG+11, NFR+17]. However, in all cases, the hardware token is significantly more complex than just implementing a PRF. Therefore, our work also provides an interesting new take on how to construct obfuscation using extremely simple hardware tokens.

Alternatively, we can interpret our result as showing that ideal obfuscation for PRFs implies ideal obfuscation for general functions. Indeed, the Pr$\mathcal{O}$ model can be thought of as exactly an ideal obfuscation for a PRF family H_k. The handle h is an ideal obfuscated program computing H_k.[3] In the literature, there are several bootstrapping theorems [GGH+13,App14,CLTV15] transforming obfuscation for weak classes of functions to that for general functions. In these works, the weak classes are typically weaker from a complexity-theoretic perspective, e.g., belonging to NC^1 or TC^0, but are expressive enough to hardcode an arbitrary circuit in the function description (e.g., verifying that a ciphertext is obtained by homomorphically evaluating a circuit on some input ciphertexts, and if so, decrypting that ciphertext). In comparison, our bootstrapping theorem starts with obfuscation of a single PRF family H_k, apparently without the ability to embed the logic related to arbitrary computation.

1.3 Further Discussion on the Pr$\mathcal{O}$ Model

The motivation behind the usual ROM is providing a rigorous and well-defined model capturing the intuition that outputs of a good hash function "appear random", and enabling formal security analysis based on this intuition. To be well-defined, the ROM completely removes non-black-box access to the hash function. Intuitively, the Pr$\mathcal{O}$ model is a new well-defined ideal functionality capturing the same intuition (as discussed earlier, it subsumes the random oracle model), and additionally allowing us to formally reason about cryptosystems that make non-black-box use of the hash function.

Formalizing *Ad Hoc* Non-Black-Box Uses of Hash Functions. The ROM was motivated and guided by heuristic uses of hash functions that preceded it (e.g., the Fiat–Shamir transformation [FS87]). However, the benefits of having an explicit ideal model go beyond providing partial justification to these usages. It greatly facilitates future design, as witnessed in the explosion of cryptosystems designed in the ROM since its introduction [BR93]. In recent years, we saw heuristic *non-black-box* uses of hash functions, for instance, in recursive composition of SNARKs [BCCT13], in simulation-secure FE [DIJ+13], and in CCA-secure FHE [CRRV17]. It is well-motivated to formalize a variant of the ROM capable of capturing some non-black-box uses of hash functions. However, prior efforts met with the contradiction that one cannot simultaneously model hash functions as random functions and assume efficient code representation. As a result, previous heuristic non-black-box uses of hash functions have been deemed less satisfactory than heuristics justified in the ROM.

The Pr$\mathcal{O}$ model side-steps the contradiction — the oracle evaluates pseudorandom

[3] This is yet another reason why the Pr$\mathcal{O}$ model and the ROM are morally equivalent. The ROM essentially says that good hash functions are "self-obfuscated PRFs" since having the full description of a hash function is no better than just having oracle access to a random function, which is also what the Pr$\mathcal{O}$ model stipulates.

functions $H_k(\cdot)$ with efficient code representation. However, as discussed earlier, security proofs in the Pr$\mathcal{O}$ model are subtle since we do not assume any security of $H_k(\cdot)$ when the key k is around. In effect, this means that the Pr$\mathcal{O}$ model only allows us to rely on RO-style modeling of $H_k(\cdot)$ in hybrid games where all non-black-box use of k is removed.

We believe that the Pr$\mathcal{O}$ model is a natural and more flexible variant of the ROM that enables us to formally reason about cryptosystems making non-black-box use of hash functions, while being "morally analogous" to the ROM.

(Non-)Contradiction of *Non-Black-Box Ideal* Hash Functions. At first sight, the idea of an ideal model capturing non-black-box use of hash functions may seem unnatural — if the cryptosystem makes non-back box use of the hash function, why can't the adversary? The Pr$\mathcal{O}$ model gives a satisfactory answer to this from two perspectives. In the formal security proof, we can only invoke Pr$\mathcal{O}$ security in hybrids where all non-black-box use of the hash function has been removed, in which case it is reasonable to assume that the adversary also only has black-box access. Conceptually, this means that the Pr$\mathcal{O}$ model is only useful for a cryptosystem if its non-black-box use of hash functions is indistinguishable to black-box use of them, so it is again reasonable to assume only black-box access from the adversary.

Putting a Real-World Cryptosystem into the Pr$\mathcal{O}$ Model. Suppose a real-world cryptosystem uses some hash function, say SHA-3 with a *public salt* k. Parts of the cryptosystem will only make black-box calls to $\mathsf{SHA3}(k\|\cdot)$, but do not rely on the code otherwise, while other parts of the cryptosystem may use the code of $\mathsf{SHA3}(k\|\cdot)$ in a non-black-box way. In the usual ROM, we model the former usage by replacing all calls to $\mathsf{SHA3}(k\|\cdot)$ by oracle calls to a truly random public oracle, but do not have any way of capturing the latter usage. In the Pr$\mathcal{O}$ model, we can set $H_k(\cdot) = \mathsf{SHA3}(k\|\cdot)$[4] and replace all black-box calls to $\mathsf{SHA3}(k\|x)$ by oracle evaluation calls on (h, x), where h is the handle to k. Additionally, if the original cryptosystem also uses $\mathsf{SHA3}(k\|\cdot)$ in a non-black-box way, the Pr$\mathcal{O}$ model allows parties knowing k to make non-black-box use of H_k and ensures that the black-box use via oracle evaluation calls is consistent with it.

The Pr$\mathcal{O}$ Paradigm. Like the ROM, the Pr$\mathcal{O}$ model articulates an explicit design paradigm. To design a good scheme or protocol P for a cryptographic notion Π:

1. Find a formal definition of Π in the model of computation where all parties (including the adversary) share the pseudorandom oracle Pr$\mathcal{O}^H$ for some PRF family H_k.
2. Devise an efficient scheme P for Π in this Pr$\mathcal{O}$ model.
3. Prove that P satisfies the definition of Π in the Pr$\mathcal{O}$ model.
4. Instantiate Pr$\mathcal{O}^H$ for some real hash function H.

In the above Pr$\mathcal{O}$ paradigm, as well as the traditional RO paradigm, the proof of security (Step 3) is in an ideal model and the instantiation (Step 4) is heuristic in

[4] We assume that $\mathsf{SHA3}(k\|\cdot)$ is a PRF with k being the *(secret) key*, which is a very mild assumption for real-world hash functions.

nature. There are known schemes/protocols secure in the ROM, but never secure when instantiated with real hash functions, e.g., [Bar01,GK03]. These counterexamples extend to the Pr$\mathcal{O}$ model. In addition, our construction of ideal obfuscation separates the Pr$\mathcal{O}$ model and the ROM. Despite these counterexamples, for the same reasons that apply to the RO paradigm, having a security proof in the Pr$\mathcal{O}$ model maintains significant benefits. First, schemes secure in the Pr$\mathcal{O}$ model are secure against generic attacks that make only black-box calls to the hash functions. Second, under the *uber heuristic* that in *natural use cases*, no adversary can effectively make use of the code of well-designed hash functions beyond making black-box calls, we obtain heuristically secure schemes in the standard model. The uber heuristic is the same as the heuristic backing the RO paradigm.

Instantiating the Pr$\mathcal{O}$ Model. Suppose some hash function H is suitable for instantiating the ROM, we argue that the salted version of H is good for the Pr$\mathcal{O}$ model. For example, to instantiate the Pr$\mathcal{O}$ model with SHA-3, we

- set $H_k(x) = \mathsf{SHA3}(k\|x)$,
- replace the handle h by k itself (!), and
- replace every evaluation call on (h, x) by $\mathsf{SHA3}(h\|x)$.

Note the glaring difference between practice and formalism. The Pr$\mathcal{O}$ model defines the handle as a random string,[5] yet the suggested instantiation simply sets $h = k$.

Our rationale is based on the random oracle heuristics — since SHA-3 is suitable for the ROM, an adversary, given the public salt $h = k$, cannot do anything meaningful about k other than evaluating it at various inputs with prefix k. This is exactly what the Pr$\mathcal{O}$ model demands from the hash function! Also, when regarding the Pr$\mathcal{O}$ model as ideal obfuscation of PRF, the handle h is the obfuscated PRF and the key k is hidden inside h, but we can redefine the PRF as $H'_h(x) = h(x) = H_k(x)$ so h becomes *the* key, i.e., it is sensible to set $h = k$. This matches the idea that hash functions suitable for the ROM are "self-obfuscated PRFs". Lastly, yet another reason to go with this simple instantiation is that the choice of making h a random string is merely a formalism, a modeling method to enable invocation of PRF security when k is absent. The means should not be taken as the ends.

1.4 Related Works

Several prior works have attempted to rely on the random oracle heuristics of a hash function while making non-black-box use of the hash function at the same time. For example, the work of Valiant [Val08] constructs incrementally verifiable proofs of knowledge and the work of [DIJ+13] constructs simulation-secure FE using this type of approach. However, these works do not define a fully specified formal model in which one can state the given results, making it difficult to even write down a meaningful theorem. This is in contrast to our Pr$\mathcal{O}$ model, which gives a formally

[5] This is similar to Shoup's generic group model [Sho97]. Alternatively, we can define the handle as a special symbol that cannot be operated on, like in Maurer's GGM [Mau05]. The two models are studied in the recent work of [Zha22]. We choose Shoup's flavor for its potential flexibility, although our construction is compatible with Maurer's. However, in either flavor, per definition, h is independent of k, and the difference between practice and formalism still prevails.

specified idealized model in which we can state and prove propositions. On the other hand, the Pr$\mathcal{O}$ model only allows very careful non-black-box use of the hash function, where we can only make use of Pr$\mathcal{O}$ security in hybrids where all non-black-box use of the hash function is removed. It does not appear that the constructions of [Val08, DIJ+13] could be directly translated into results in the Pr$\mathcal{O}$ model.

2 Technical Overview

Now we describe the main ideas behind our construction. Our starting point is the insights of [BV15,AJ15] and the follow-ups [LPST16,BNPW16,KNT18,KNTY19], which establish that $i\mathcal{O}$ can implemented generically from subexponentially secure single-key functional encryption (FE) scheme, a seemingly weaker primitive. These works additionally require the FE scheme to satisfy certain encryption efficiency guarantees. In the overview below, we assume that the FE scheme satisfies adaptive indistinguishability security and has linear-time encryption.[6]

FE-to-$i\mathcal{O}$ Transformation. In order to obfuscate circuit $C : \{0,1\}^D \to \{0,1\}$, we give out an FE ciphertext ct_ε encrypting C. We think of this ciphertext as being associated with the root of a perfect binary tree of depth D. We also give out FE secret keys for each of the D levels in the tree, for functions that themselves compute FE encryptions. By defining such functions carefully, we can expand any ciphertext ct_χ for some prefix $\chi \in \{0,1\}^{<D}$ associated with some internal node in the tree into two ciphertexts $\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1}$ for its children, with each such child ciphertext again carrying information about C. Lastly, for leaf ciphertexts ct_x with $x \in \{0,1\}^D$, we give out an FE secret key that allows one to recover the output $C(x)$. This allows an evaluator to compute $C(x)$ starting from ct_ε by going down the appropriate path in the tree.

In more detail, the obfuscator does the following:

- For $0 \le d \le D$, sample fresh FE key pairs $(\mathsf{mpk}_d, \mathsf{msk}_d)$.
- Compute $\mathsf{ct}_\varepsilon \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}_0, \mathsf{info}_\varepsilon)$, where $\mathsf{info}_\varepsilon = (C, \varepsilon, \star)$ with $\star$ being a slot for miscellaneous information to be specified later as needed.
- For $0 \le d \le D$, generate $\mathsf{sk}_d \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}_d, f_d)$. Under normal functioning, f_d for $i < D$ takes $\mathsf{info}_\chi = (C, \chi, \star)$ as input and outputs two ciphertexts $(\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1})$ encrypting $\mathsf{info}_{\chi\|b} = (C, \chi\|b, \star)$ for $b \in \{0,1\}$ under mpk_{d+1}, and f_D takes $\mathsf{info}_x = (C, x \in \{0,1\}^D, \star)$ as input and outputs $C(x)$.

The obfuscated circuit is $\widehat{C} = (\mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\le d\le D})$. To evaluate $\widehat{C}$ on $x \in \{0,1\}^D$, one computes ct_x at level D, then decrypts it using sk_D to recover $C(x)$. The process of computing ct_x is inductive and proceeds like a binary tree traversal. Let $x_{\le d}$ be the prefix of x of length d. We start by decrypting $\mathsf{ct}_{x_{\le 0}} = \mathsf{ct}_\varepsilon$ using sk_0 to obtain $(\mathsf{ct}_0, \mathsf{ct}_1)$. For $1 \le d < D$, we inductively decrypt $\mathsf{ct}_{x_{\le d}}$ using sk_d to derive $(\mathsf{ct}_{x_{\le d}\|0}, \mathsf{ct}_{x_{\le d}\|1})$, then use $\mathsf{ct}_{x_{\le d+1}}$ to further traverse down the tree.

The scheme satisfies *polynomial slowdown*. At every level, ct_χ encrypts info_χ and the running time of f_d is $\mathrm{poly}(\lambda, |\mathsf{info}_\chi|)$, so each f_d is polynomial-sized. Some care is

[6] The encryption time is $|z|\,\mathrm{poly}(\lambda)$, where z is the plaintext. This is independent of the functions for which secret keys are issued.

needed to ensure that the $\star$ slots used by the ciphertexts do not blow up as the levels increase — the proof is designed in a way that this happens.

The security proof is slightly tricky. Given the obfuscation, there are around 2^D FE ciphertexts, ct_χ for $\chi \in \{0,1\}^{\leq D}$, encrypting $\mathsf{info}_\chi = (C, \chi, \star)$, containing the circuit C being obfuscated. For two equivalent circuits C_0, C_1, we want to show $\widehat{C}_0 \approx \widehat{C}_1$. Since the adversary is given ct_ε and $\mathsf{sk}_0, \ldots, \mathsf{sk}_D$, it can compute ct_χ for any $\chi \in \{0,1\}^{\leq D}$ of its choice, and can do so internally without the reduction knowing the χ's "of the adversary's interest". Therefore, the proof resorts to switching ct_χ from containing C_0 to C_1 for *every* $\chi \in \{0,1\}^{\leq D}$, taking around 2^D steps (hybrids) and creating a security loss of $\Omega(2^D)$.

Why the Scheme Fails to Be an Ideal Obfuscation. In ideal obfuscation, we require that $\widehat{C}$ can be simulated by a *polynomial-time* simulator having only oracle access to C. This implies that the simulator must come up with a short ciphertext ct_ε (ignoring the [short] keys for now) capable of evaluating C at every $x \in \{0,1\}^D$ from just black-box access to C. Assuming that hard-to-learn functions exist, this is impossible as the simulator can only query C at a polynomial number of points. Indeed, this simple argument shows that ideal obfuscation cannot exist, which is also implied by the the work of [BGI+01] showing that even a more restricted notion, VBB obfuscation, cannot exist. In this work, one of our primary goals is to identify a reasonable model capturing real-world adversaries in which ideal obfuscation is possible. We take inspiration from the random oracle model [BR93] — there are several applications known to be impossible in the standard model but achievable in the ROM.

Simplified Idea Using Random Oracles. From the observation discussed earlier, one of the obstacles to proving ideal security of the scheme above is that once the reduction or the simulator produces some ciphertext ct_ε, it has implicitly specified *all* the ciphertexts in the tree and all the outputs of the circuit. There is no place to "program" any information. The random oracle could be useful in solving this issue. Imagine a world where ct_ε makes the first decryption yield $H(\varepsilon) \oplus (\mathsf{ct}_0 \| \mathsf{ct}_1)$ so that adversary has to query $H(\varepsilon)$ to unmask the next layer ciphertexts. Similarly, assume that the result of decrypting ct_χ for $\chi \in \{0,1\}^{\leq D-1}$ is $H(\chi) \oplus (\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1})$.

If the above were possible, we would be able to come up with a simulation strategy. The random oracle offers two powerful capabilities, *observability* and *programmability*, that enable such simulation. As the evaluator queries H at various χ, the simulator can keep track of the paths the adversary is taking to evaluate the circuit. By programming, one can enter a hybrid where ct_χ decrypts to a random value v_χ and (simultaneously) the random oracle responds to $H(\chi)$ by answering $v_\chi \oplus (\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1})$.

Once this is done, any ct_χ can only be accessed by querying the oracle. The issue of programming space would be resolved as the random oracle provides an exponentially large one. In addition, since the adversary only makes polynomially many queries, the proof only has to switch a polynomial number of ciphertexts, reducing the security loss to polynomial. In particular, the proof would replace ct_χ for $\chi \in \{0,1\}^{\leq D-1}$ by dummy ciphertexts independent of C, and ct_x for $x \in \{0,1\}^D$ by simulated ciphertexts containing $C(x)$, *only* for χ's and x's at which H is queried.

Unfortunately, so far this is just wishful thinking! There is a fundamental flaw with the above idea that must be addressed before we can materialize this approach.

Using the Pr$\mathcal{O}$ Model. The flaw with the aforementioned idea is the premise itself. We assume that decrypting ct_χ yields $H(\chi) \oplus (\cdots)$, but this requires the FE scheme to evaluate the hash function H. This makes sense only if the FE key functions f_d contain the code of H, meaning that H must be a real hash function and not a random oracle! We now have seemingly conflicting requirements — we need the code of the hash function to define the scheme syntactically, but we also need to idealize the hash function as a public random function to take advantage of observability and programmability.

This is where our model comes in. We precisely show that the above approach can be done in the Pr$\mathcal{O}$ model. Let us briefly recall the model. There are two oracle algorithms hGen and hEval with syntax

$$\mathcal{O}(\mathsf{hGen}, k) \mapsto h, \quad \mathcal{O}(\mathsf{hEval}, h, x) \mapsto H(k, x),$$

where $\mathcal{O}(\mathsf{hGen}, \cdot)$ maps a key k into a handle h and $\mathcal{O}(\mathsf{hEval}, h, x)$ maps the handle h back into its key k and outputs $H(k, x)$ for some fixed function H. The handle map is a random permutation and can be efficiently implemented using lazy sampling. Furthermore, we require that H is a pseudorandom function — this implies that given a handle h for a random k, when k is *absent* from the view of the adversary, $\mathcal{O}(\mathsf{hEval}, h, \cdot) = H(k, \cdot)$ is indistinguishable from a random function.

The Pr$\mathcal{O}$ model provides the right abstraction needed to solve our problem. A random key k can be used *inside* the FE ciphertext, together with the code of H in the FE keys, to compute $H(k, \chi) \oplus (\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1})$, and the corresponding handle h can be used *outside* the FE scheme when evaluating the obfuscated circuit. At the same time, if k is absent from the adversary's view, we can still program $\mathcal{O}(\mathsf{hEval}, h, \cdot)$. Of course, the difficulty is that the key k is part of the adversary's view whenever it is (encrypted) inside the FE ciphertext. Therefore, we must come up with a careful proof strategy involving a sequence of hybrids to first remove k then program the oracle. We explain how to do so below.

First Attempt. Let $G : \{0,1\}^\lambda \to \{0,1\}^{4\lambda}$ be a PRG for deriving the encryption randomness of the intermediate FE ciphertexts. Here, we assume that both Pr$\mathcal{O}$ hash keys and encryption randomness are of length λ. To obfuscate C, one computes ct_ε encrypting $(C, \varepsilon, k, s_\varepsilon)$, where k is a fresh hash key (with h being its handle) and s_ε is a fresh PRG seed. The function f_0 on input $(C, \varepsilon, k, s_\varepsilon)$ outputs $H(k, \varepsilon) \oplus (\mathsf{ct}_0 \| \mathsf{ct}_1)$, and $f_1, \ldots, f_D$ are defined analogously. The obfuscator outputs $\widehat{C} = (h, \mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0 \le d \le D})$. We give an outline in Figure 1.

Evaluating such an obfuscated circuit is straightforward. Given $x \in \{0,1\}^D$, for each prefix χ of x starting from ε, decrypting ct_χ yields the child ciphertexts masked by $H(k, \chi)$, which the evaluator can unmask by querying $(\mathsf{hEval}, h, \chi)$ before proceeding to the next level. Lastly, decrypting ct_x yields $C(x)$.

As envisioned, the simulator programs $\mathcal{O}(\mathsf{hEval}, h, \cdot)$ and changes the ciphertexts into some simulation mode. A natural idea to prove security is to replace the (intermediate) ciphertexts layer by layer in a sequence of hybrids, starting from the root ct_ε. This approach faces an immediate obstacle. In the Pr$\mathcal{O}$ model, $\mathcal{O}(\mathsf{hEval}, h, \cdot)$ can only be programmed if the key k is not given to the adversary, but in the scheme, k appears inside all the intermediate ciphertexts. Therefore, it is not clear how to switch even just ct_ε into simulation, as k appears in the deeper ciphertexts in both hybrids. We would have to first remove k from all the intermediate ciphertexts to

appeal to the programmability of $\mathcal{O}(\mathsf{hEval}, h, \cdot)$. This appears rather difficult without first simulating the ciphertexts by programming, i.e., there is a circularity issue.

Input. C, the circuit being obfuscated.

Steps.

1. Sample $k \xleftarrow{\$} \{0,1\}^\lambda$ and obtain its handle h from $\Pr\mathcal{O}$.
2. Sample $s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
3. For $0 \le d \le D$, sample fresh FE key pair $(\mathsf{mpk}_d, \mathsf{msk}_d)$ and run $\mathsf{sk}_d \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}_d, f_d)$ for f_d defined below.
4. Run $\mathsf{ct}_\varepsilon \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}_0, \mathsf{info}_\varepsilon)$ with $\mathsf{info}_\varepsilon = (\mathsf{normal}, \varepsilon, k, s_\varepsilon)$. Here, normal is a flag indicating the mode of the ciphertext.
5. Output $\widehat{C} = (h, \mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\le d\le D})$.

Functions. (normal mode)

– For $0 \le d < D$, the function $f_d(\mathsf{normal}, \chi, k, s_\chi)$
  1. computes $s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|0}\|r_{\chi\|0} \leftarrow G(s_\chi)$;
  2. for $b \in \{0,1\}$, runs $\mathsf{ct}_{\chi\|b} \leftarrow \mathsf{Enc}(\mathsf{mpk}_{d+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$, where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, k, s_{\chi\|b})$;
  3. outputs $H(k,\chi) \oplus (\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1})$.

– For $d = D$, the function $f_D(\mathsf{normal}, C, x, k, s_x)$ outputs $C(x)$.

Figure 1. First attempt of ideal obfuscator.

Second Attempt. To circumvent the issue, we use different key/handle pairs (h_i, k_i) for each layer as opposed to the same pair. Now, a ciphertext ct_χ for $\chi \in \{0,1\}^d$ at level d will only contain $\{(h_i, k_i)\}_{d\le i<D}$. Note that each ciphertext is independent of the keys for the previous layers, thus breaking the circularity. The scheme is described in Figure 2.

The simulator now programs $\mathcal{O}(\mathsf{hEval}, h_d, \cdot)$'s accordingly. Our proof strategy is again hybridizing over the layers. Suppose the ciphertexts ct_χ for $\chi \in \{0,1\}^{<\delta}$ are already simulated so that they contain no information about $C, k_0, \ldots, k_{\delta-1}$, and we want to simulate ct_χ's for $\chi \in \{0,1\}^\delta$. Now that k_δ only appears in ct_χ's for $\chi \in \{0,1\}^\delta$ and nowhere else, it is easier to completely remove k_δ, at which point we can observe and program $\mathcal{O}(\mathsf{hEval}, h_\delta, \cdot)$ and replace ct_χ's by dummy ciphertexts.

Take $\delta = 0$ as an example, for which our goal is to change ct_ε from encrypting $(\mathsf{normal}, C, \varepsilon, \{k_i\}_{0\le i<D}, s_\varepsilon)$ to being simulated and to program $\mathcal{O}(\mathsf{hEval}, h_0, \cdot)$. This can be done in three steps:

1. Change ct_ε to encrypting $(\mathsf{sim}, H(k_0,\varepsilon) \oplus (\mathsf{ct}_0\|\mathsf{ct}_1))$. Seeing the sim flag, the function f_0 simply outputs the second component, the *hardwired* output. This change is indistinguishable to the security of FE.

2. Replace the appearance of $H(k_0,\varepsilon)$ in ct_ε and the response to $\mathcal{O}(\mathsf{hEval}, h_0, \varepsilon)$ by a random string otp_ε. This change is indistinguishable by the PRF security of H.

3. Change ct_ε to encrypting $(\mathsf{sim}, v_\varepsilon)$ and replace the response to $\mathcal{O}(\mathsf{hEval}, h_0, \varepsilon)$ by $v_\varepsilon \oplus (\mathsf{ct}_0\|\mathsf{ct}_1)$, where v_ε is a random string. This is identical to the previous hybrid.

After these steps, we are ready to work with $\mathsf{ct}_0, \mathsf{ct}_1$. This strategy generalizes to every layer of the tree, except that each leaf ct_x will hardwire $C(x)$ instead of a random v_x during simulation. Moreover, the simulator only creates $\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1}$ when the adversary queries $\mathcal{O}(\mathsf{hEval}, h_{|\chi|}, \chi)$, and only queries $C(x)$ if it has to create ct_x. This implies that the simulator is polynomial-time and the security loss of the proof is polynomial.

While the intuition is simple, there is still an important issue that we overlooked. The problem is that the *plaintext* inside ct_χ is of length $\left|\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}\right|$, *twice* as long as a (one-layer-deeper) *ciphertext*. As such, $|\mathsf{ct}_\varepsilon|$ grows at least exponentially as D increases. To resolve this issue, we revisit the simulation strategy at the very first layer and then apply the idea to the all layers.

Input. C, the circuit being obfuscated.

Steps.

1. For $0 \le i < D$, sample $k_i \xleftarrow{\$} \{0,1\}^\lambda$ and obtain its handle h_i from $\Pr\mathcal{O}$.
2. Sample $s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
3. For $0 \le d \le D$, sample fresh FE key pair $(\mathsf{mpk}_d, \mathsf{msk}_d)$ and run $\mathsf{sk}_d \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}_d, f_d)$ for f_d defined below.
4. Run $\mathsf{ct}_\varepsilon \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}_0, \mathsf{info}_\varepsilon)$ with $\mathsf{info}_\varepsilon = (\mathsf{normal}, \varepsilon, \{k_i\}_{0\le i<D}, s_\varepsilon)$.
5. Output $\widehat{C} = (\{h_i\}_{0\le i<D}, \mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\le d\le D})$.

Functions. (normal mode)

- For $0 \le d < D$, the function $f_d(\mathsf{normal}, \chi, \{k_i\}_{d\le i<D}, s_\chi)$
  1. computes $s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|0}\|r_{\chi\|0} \leftarrow G(s_\chi)$;
  2. for $b \in \{0,1\}$, runs $\mathsf{ct}_{\chi\|b} \leftarrow \mathsf{Enc}(\mathsf{mpk}_{d+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$, where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, \{k_i\}_{d+1\le i<D}, s_{\chi\|b})$;
  3. outputs $H(k_d, \chi) \oplus (\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1})$.
- For $d = D$, the function $f_D(\mathsf{normal}, C, x, s_x)$ outputs $C(x)$. (There are no k's in the leaf ciphertexts.)

Figure 2. Second attempt of ideal obfuscator.

Fixing Simulation Efficiency. To fix the simulator itself, instead of a long, truly random v_ε, we encrypt a short PRG seed in ct_ε and let f_0 output the PRG expansion during simulation. We must also not hardwire the complete decryption result into ct_ε in Step 1 above. The solution is to work on the decryption result block by block. The scheme is outlined in Figure 3.

To demonstrate that, let $|\mathsf{ct}|$ be (an upper bound of) FE ciphertext length. We define two parameters B, the number of blocks, and L, the length of each block, which we set so that $LB \ge 2|\mathsf{ct}|$. Now, instead of one key/handle pair (h_0, k_0) for the first layer, we use B pairs $\{(h_{0,j}, k_{0,j})\}_{1\le j\le B}$. We also assume $H(k, \cdot)$ is of length L. To switch ct_ε from real to simulation, consider the following series of hybrids for $1 \le \beta \le B$:

1. Remove $k_{0,\beta}$ from ct_ε and hardwire $H(k_{0,\beta}, \varepsilon) \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]_\beta$, the β^{th} block of the output, into ct_ε. Here, $[\cdot]_\beta$ is the β^{th} block of a string.

2. Replace the hardwired block in ct_ε by random and program $\mathcal{O}(\mathsf{hEval}, h_{0,\beta}, \varepsilon)$ for consistency. This is allowed since $k_{0,\beta}$ has been removed from the view of the adversary.

3. Replace the hardwired block in ct_ε by PRG expansion (while reprogramming $\mathcal{O}(\mathsf{hEval}, h_{0,\beta}, \varepsilon)$ for consistency). This change is indistinguishable due to the security of PRG.

4. Undo hardwiring by putting the PRG seed into ct_ε.

This idea can be generalized to every node in every layer, except that the last layer is handled using $C(x)$. The hybrid argument will proceed layer by layer, and inside each layer, block by block. For each block, all the (queried) ciphertexts are switched together.

To see that the lengths are under control, note that the plaintext contains (at most) the circuit, an input prefix, DB hash keys, one L-bit block, and B PRG seeds. Recall that we assume linear-time encryption in this overview, so $|\mathsf{ct}| = \mathsf{O}(|C| + D + DB\lambda + L + B\lambda)$. The constraint $LB \geq 2|\mathsf{ct}|$ can be easily satisfied by setting $L = B = \Theta(|C|D\lambda)$.

Input. C, the circuit being obfuscated.

Steps.

1. For $0 \leq i < D, 1 \leq j \leq B$, sample $k_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$ and obtain its handle $h_{i,j}$ from $\Pr\mathcal{O}$.
2. Sample $s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
3. For $0 \leq d \leq D$, sample fresh FE key pair $(\mathsf{mpk}_d, \mathsf{msk}_d)$ and run $\mathsf{sk}_d \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{msk}_d, f_d)$ for f_d defined below.
4. Run $\mathsf{ct}_\varepsilon \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}_0, \mathsf{info}_\varepsilon)$ with $\mathsf{info}_\varepsilon = (\mathsf{normal}, \varepsilon, \{k_{i,j}\}_{\substack{0\leq i<D \\ 1\leq j\leq B}}, s_\varepsilon)$.
5. Output $\widehat{C} = (\{h_{i,j}\}_{\substack{0\leq i<D \\ 1\leq j\leq B}}, \mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\leq d\leq D})$.

Functions. (normal mode)

– For $0 \leq d < D$, the function $f_d(\mathsf{normal}, \chi, \{k_{i,j}\}_{\substack{d\leq i<D \\ 1\leq j\leq B}}, s_\chi)$
  1. computes $s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|0}\|r_{\chi\|0} \leftarrow G(s_\chi)$;
  2. for $b \in \{0,1\}$, runs $\mathsf{ct}_{\chi\|b} \leftarrow \mathsf{Enc}(\mathsf{mpk}_{d+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$, where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, \{k_{i,j}\}_{\substack{d+1\leq i<D \\ 1\leq j\leq B}}, s_{\chi\|b})$;
  3. outputs $\big(H(k_{d,1}, \chi)\|\cdots\|H(k_{d,B}, \chi)\big) \oplus (\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1})$.

– For $d = D$, the function $f_D(\mathsf{normal}, C, x, s_x)$ outputs $C(x)$. (There are no k's in the leaf ciphertexts.)

Figure 3. Final construction of ideal obfuscator.

3 Preliminaries

We denote by λ the security parameter, and use the standard notions $\approx, \approx_s, \equiv$ for computational indistinguishability, statistical indistinguishability, and identity. The

order of tuples of ordered objects is lexicographical, so $(a, b) \leq (c, d)$ means either $a = c$ and $b \leq d$ or $a < c$, for integers a, b, c, d. For two strings x, y, we write $x \| y$ for their concatenation. The empty string is denoted by ε. Given a string x and a length $0 \leq i \leq |x|$, we let $x_{\leq i}$ be the length-i prefix of x. In the context where strings are canonically split into blocks, $[x]_j$ denotes the j^{th} block of x for $j \geq 1$. For a circuit C, we write $C[w]$ for C with w hardwired into its leading portion of input.

Pseudorandom Generators and Pseudorandom Functions. We assume that the PRG seed length is always λ, that its output length ℓ_{out} can be freely specified, and that its running time is $\ell_{\text{out}} \operatorname{poly}(\lambda)$. Similarly, we assume that the PRF key is uniformly random over $\{0,1\}^\lambda$, that its input/output lengths $\ell_{\text{in}}, \ell_{\text{out}}$ can be freely specified, and that its running time is $\ell_{\text{in}} \ell_{\text{out}} \operatorname{poly}(\lambda)$.[7]

Functional Encryption. We base our obfuscation scheme on 1-key functional encryption, which is weaker than the standard notion:[8]

Definition 1 (1-key FE [BV15])**.** A *(public-key) 1-key functional encryption scheme (for circuits)* consists of 3 efficient algorithms:

- $\mathsf{Gen}(1^\lambda, f)$ takes a circuit $f : \{0,1\}^n \to \{0,1\}^*$ as input. It outputs a pair $(\mathsf{pk}, \mathsf{sk}_f)$ of public (encryption) key and secret (decryption) key for f.
- $\mathsf{Enc}(\mathsf{pk}, z)$ takes as input the public key and some plaintext $z \in \{0,1\}^n$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$ takes as input the secret key and a ciphertext. It is supposed to compute $f(z)$.

The scheme must be *correct,* i.e., for all $\lambda \in \mathbb{N}$, circuit $f : \{0,1\}^n \to \{0,1\}^*$, input $z \in \{0,1\}^n$, it holds that

$$\Pr\left[\begin{array}{c}(\mathsf{pk}, \mathsf{sk}_f) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, f) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, z)\end{array} : \mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) = f(z)\right] = 1.$$

We require the encryption algorithm to run in time subquadratic in $|z|$ and sublinear in $|f|$:

Definition 2 (efficiency)**.** A 1-key FE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (Definition 1) has *subquadratic-sublinear efficiency* (or *sufficiently efficient* for the purpose of this work) if Enc runs in time

$$(n^{2-2\varepsilon} + m^{1-\varepsilon}) \operatorname{poly}(\lambda) \qquad \text{for some constant } \varepsilon > 0,$$

where $n = |z|$ is the input length of f and $m = |f|$ is the circuit size of f.

By a standard result [PF79] in circuit complexity, a circuit of Enc of subquadratic-sublinear size can be efficiently computed. Hereafter, we will use such a bound for uniform circuit complexity of Enc.

[7] The required properties can be achieved by standard PRG extension techniques and indifferentiable domain extension of random oracles.

[8] In retrospect, this notion is an interpolation between functional encryption and unary function-revealing encryption [JP18].

We need the 1-key FE scheme to be *adaptively* secure:

Definition 3 (adaptive security)**.** A 1-key FE scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (Definition 1) is *adaptively secure* if $\mathsf{Exp}^0_{\text{1-key}} \approx \mathsf{Exp}^1_{\text{1-key}}$, where $\mathsf{Exp}^b_{\text{1-key}}(1^\lambda)$ with adversary $\mathcal{A}$ proceeds as follows:

- **Setup.** Launch $\mathcal{A}(1^\lambda)$, receive a circuit $f : \{0,1\}^n \to \{0,1\}^*$ from $\mathcal{A}$, run

$$(\mathsf{pk}, \mathsf{sk}_f) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, f),$$

  and send $(\mathsf{pk}, \mathsf{sk}_f)$ to $\mathcal{A}$.

- **Challenge.** $\mathcal{A}$ chooses two inputs $z_0, z_1 \in \{0,1\}^n$. Run $\mathsf{ct} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, z_b)$ and send ct to $\mathcal{A}$.

- **Guess.** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The outcome of the experiment is b' if $f(z_0) = f(z_1)$. Otherwise, the outcome is set to 0.

There is a long series of works [AJS15,ABSV15,BV15,GS16,LM16,AS16,KNTY19,JLL23] studying the transformations among functional encryption schemes with various security and efficiency guarantees. It is known [JLL23,Nis22] that standard public-key FE with encryption time $m^{1-\varepsilon}\,\mathrm{poly}(\lambda, n)$ and weak selective security against one key query implies standard public-key FE with encryption time $n\,\mathrm{poly}(\lambda)$ and full adaptive security against unbounded collusion.[9] The latter can be used as a sufficiently efficient and adaptively secure 1-key FE.

We can assume, without loss of generality (neither efficiency nor security), that Enc uses a uniformly random λ-bit string as its randomness, by using a PRG with efficiency stated earlier in this section.

Idealized Model. We will define ideal obfuscation with respect to an idealized model, and construct such a scheme in a particular idealized model.

Definition 4 (idealized model)**.** In an *idealized model* with oracle $\mathcal{O}$, all algorithms, including adversaries, are given access to $\mathcal{O}$. The oracle is programmable, i.e., security reductions as well as simulators in simulation-based security notions can provide an alternative implementation of $\mathcal{O}$.

As an example, the standard model is an idealized model with $\mathcal{O}() = \bot$.

Oracle Circuits. In an idealized model, we may consider circuits containing gates calling into $\mathcal{O}$, referred to as *oracle circuits*. Like a usual circuit, the description $C^\bullet$ of an oracle circuit consists of its gates and wires, with the convention that oracle gates are just placeholders, i.e., $C^\bullet$ does not specify the behavior of the oracle. The circuit can be evaluated given an input x and an oracle $\mathcal{O}$ (with appropriate input/output lengths), which is denoted by $C^{\mathcal{O}}(x)$.

[9] In a standard public-key FE, the scheme is set up for a master public/secret key pair not tied to f, and a key for f can be derived separately from the master secret key. Weak selective security means that the adversary chooses f, z_0, z_1 independent of the master public key. Full adaptive security against unbounded collusion means that the adversary can choose z_0, z_1 and arbitrarily many f_q's after seeing the master public key and in an arbitrary interleaving manner.

4 The Pseudorandom Oracle (Pr$\mathcal{O}$) Model

We now formally define the pseudorandom oracle model.

Definition 5 (Pr$\mathcal{O}$M)**.** Let H be a pseudorandom function. The *pseudorandom oracle model* for H is the idealized model with the oracle $\mathcal{O}$ that internally uses a random permutation

$$\mathsf{hMap} : \{0,1\}^\lambda \to \{0,1\}^\lambda$$

and that responds to the following 2 types of queries:

$$\mathcal{O}(\mathsf{hGen}, k) = \mathsf{hMap}(k), \quad \mathcal{O}(\mathsf{hEval}, h, t) = H(\mathsf{hMap}^{-1}(h), t).$$

5 Ideal Obfuscation

We still define ideal obfuscators as algorithms and security properties in idealized models, instead of an idealized model itself. The reason is elaborated after the security definition.

Definition 6 ((circuit) obfuscation)**.** A *(circuit) obfuscation scheme* in an idealized model with oracle $\mathcal{O}$ is an efficient algorithm $\mathsf{Obf}^{\mathcal{O}}(1^\lambda, C)$ that, given a circuit C as input, outputs an oracle circuit $\widehat{C}^\bullet$. The scheme must be *correct*, i.e., for all $\lambda \in \mathbb{N}$, circuit $C : \{0,1\}^D \to \{0,1\}^*$, input $x \in \{0,1\}^D$, it holds that

$$\Pr\left[\, \widehat{C}^\bullet \xleftarrow{\$} \mathsf{Obf}^{\mathcal{O}}(1^\lambda, C) \;:\; \widehat{C}^{\mathcal{O}}(x) = C(x) \,\right] = 1.$$

We remark that the scheme can only obfuscate *vanilla* circuits, which do not use the idealized model oracle $\mathcal{O}$, yet the oracle $\mathcal{O}$ can be used during evaluation. This gap is necessary to avoid the impossibility results [BGI+01].

Our definition of ideal obfuscation in an idealized model is inspired by the indifferentiability framework [MRH04]:

Definition 7 (ideal obfuscation)**.** An obfuscation scheme $\mathsf{Obf}^{\mathcal{O}}$ (Definition 6) is an *ideal obfuscation (with universal simulation)* if there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ (with shared state) such that for all efficient adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ (with shared state), its advantage is negligible:

$$\Pr\left[\begin{matrix} C \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}}(1^\lambda) \\ \widehat{C}^\bullet \xleftarrow{\$} \mathsf{Obf}^{\mathcal{O}}(1^\lambda, C) \end{matrix} \;:\; \mathcal{A}_2^{\mathcal{O}}(\widehat{C}^\bullet) = 1\right] - \Pr\left[\begin{matrix} C \xleftarrow{\$} \mathcal{A}_1^{\mathcal{S}_1}(1^\lambda) \\ \widetilde{C}^\bullet \xleftarrow{\$} \mathcal{S}_2^{C}(1^\lambda, 1^D, 1^S) \end{matrix} \;:\; \mathcal{A}_2^{\mathcal{S}_3^{C}}(\widetilde{C}^\bullet) = 1\right].$$

Here, $D = |x|$ is the input length of C, and $S = |C|$ is the circuit size of C.

Although Definition 7 does not guarantee security when multiple circuits are obfuscated, the simulator for our construction readily extends to handle multiple circuits.

We add a remark about our security definition and the ideal obfuscation model alluded to in the introduction. The idealized model $\mathsf{IdealObf}$ is as follows:

- $\mathsf{IdealObf}(\mathsf{hGen}, C)$ takes a circuit $C : \{0,1\}^n \to \{0,1\}$ as input and outputs a handle h.

- $\mathsf{IdealObf}(\mathsf{hEval}, h, x)$ takes as input a handle h and some $x \in \{0,1\}^n$, and outputs $C(x)$, where C is the circuit corresponding to h.

The usual way of defining that Obf is a construction of IdealObf in (or from) $\mathcal{O}$ via indifferentiability [MRH04] is to require

$$(\mathsf{Obf}^{\mathcal{O}}, \mathcal{O}) \approx (\mathsf{IdealObf}, \mathcal{S}^{\mathsf{IdealObf}})$$

for some (potentially distinguisher-dependent) simulator $\mathcal{S}$. In this framework, the distinguisher communicates directly with IdealObf. The simulator only plays the role of $\mathcal{O}$ without observing or controlling the interaction between the distinguisher and IdealObf. In contrast, in Definition 7, the simulator gets partial information of the circuit being obfuscated (namely, its size and its input length) and simulates its obfuscation, in addition to handling $\mathcal{O}$-queries. The reason we do not use the language of indifferentiability is as follows. Unlike hash functions (often considered for indifferentiability), whose desired output is just a random string, an obfuscated circuit is a structured object, whose format has no *a priori* specification and is inherently obfuscator-dependent.[10] In addition, the simulator must somehow learn the unavoidable leakage of circuit size and input length. This makes it cumbersome, if possible at all, to formulate the definition as indifferentiability. Nevertheless, we consider our definition essentially indifferentiability with minimal and necessary "interface gluing".

6 Construction of Ideal Obfuscation in the Pr$\mathcal{O}$ Model

Similar to many prior works [AJ15,BV15] building obfuscation from FE, our scheme involves a binary tree of FE ciphertexts, with certain difference to take advantage of the Pr$\mathcal{O}$ model. The binary tree structure is instructive for understanding the correctness, as well as the security proof in Section 7, of our construction.

The obfuscation of a circuit C with D-bit input involves a perfect binary tree of $(D+1)$ levels, as depicted in Figure 4. Each node is identified by its root-to-node path, each leaf an input x to C, and each internal node a proper prefix of x. For each $\chi \in \{0,1\}^{\le D}$, node χ is associated with ct_χ encrypting C, χ plus some other information. The behavior of decrypting ct_χ is as follows:

- For an internal node, $\chi \in \{0,1\}^{<D}$ is a proper fix of the input, and decrypting ct_χ yields its children $\mathsf{ct}_{\chi\|0}$ and $\mathsf{ct}_{\chi\|1}$ *padded* by the one-time pad otp_χ associated with χ, which is the Pr$\mathcal{O}$M oracle output.
- For a leaf, $\chi = x \in \{0,1\}^D$ is the input, and decrypting ct_x yields $C(x)$.

The obfuscated circuit $\widehat{C}^\bullet$ contains the root ciphertext ct_ε, FE secret keys, and handles of Pr$\mathcal{O}$M. To evaluate $C(x)$, starting from the root ciphertext ct_ε, for each proper prefix χ of x, we decrypt ct_χ, *unpad* the result using otp_χ, and keep either $\chi_{\chi\|0}$ or $\mathsf{ct}_{\chi\|1}$ (depending on the next bit of x), until we reach ct_x, which we decrypt one last time for $C(x)$.

[10] Suppose Obf_1 is an ideal obfuscator, then Obf_2 that adds a dummy gate to the output of Obf_1 should also be considered ideal. However, the outputs of the two obfuscators are clearly distinguishable. Therefore, they cannot both be indifferentiable implementations of a single ideal obfuscation model, i.e., no fixed IdealObf (with fixed format of handle) can comprehensively capture ideal obfuscation in the indifferentiability framework. To cast our definition as indifferentiability, it is necessary to consider a different IdealObf for each obfuscator.

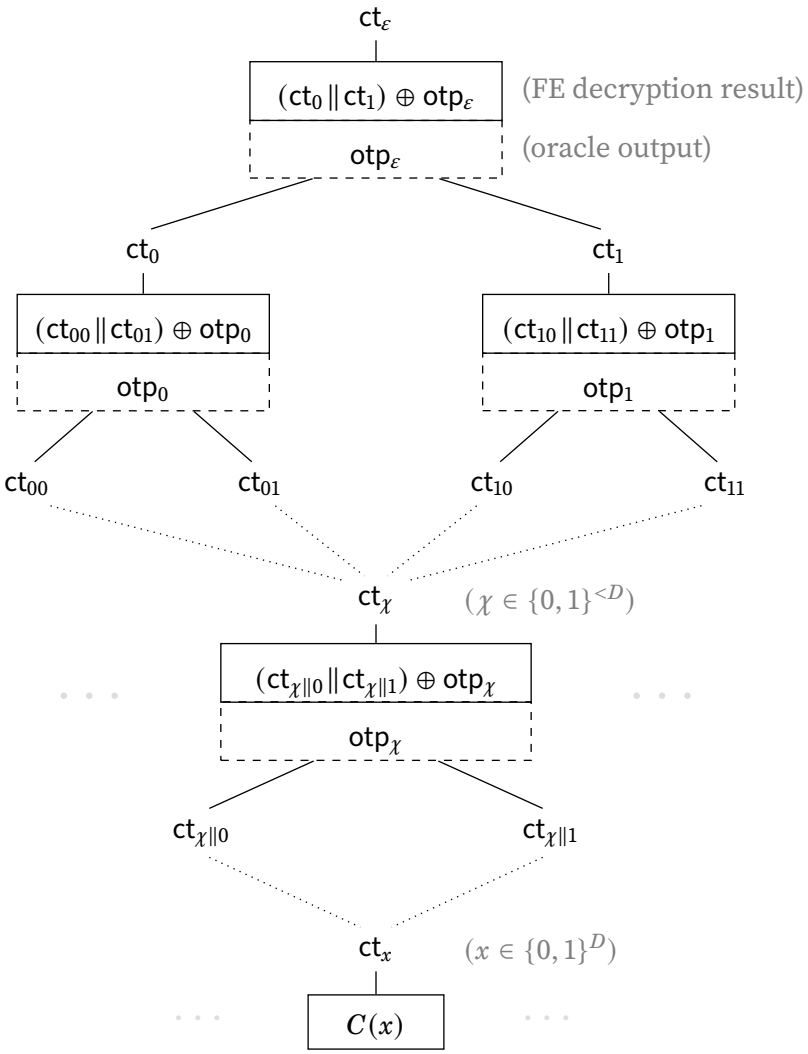

Figure 4. The binary tree of ciphertexts in Construction 1 (normal behavior).

$\mathsf{Expand}_d[\mathsf{pk}_{d+1}](\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi)$ — **Function for Level** $0 \le d < D$

Hardwired. pk_{d+1}, public key for level $(d+1)$.

Input. $\mathsf{flag}_\chi \in \{\mathsf{normal}, \mathsf{hyb}, \mathsf{sim}\}$, flag associated with χ;
$\chi \in \{0,1\}^d$, proper prefix of circuit input;
info_χ, information associated with χ, format varying by flag_χ.

Output.
$$\left\{\begin{array}{ll} \mathsf{Expand}_{d,\mathsf{normal}}[\mathsf{pk}_{d+1}](\chi, \mathsf{info}_\chi), & \text{if } \mathsf{flag}_\chi = \mathsf{normal}; \\ \mathsf{Expand}_{d,\mathsf{hyb}}[\mathsf{pk}_{d+1}](\chi, \mathsf{info}_\chi), & \text{if } \mathsf{flag}_\chi = \mathsf{hyb}; \\ \mathsf{Expand}_{d,\mathsf{sim}}(\chi, \mathsf{info}_\chi), & \text{if } \mathsf{flag}_\chi = \mathsf{sim}. \end{array}\right\} \text{Figure 7}$$

$\mathsf{Eval}(\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi)$ — **Function for Level** D

Input. $\mathsf{flag}_\chi \in \{\mathsf{normal}, \mathsf{sim}\}$, flag associated with χ;
$\chi \in \{0,1\}^D$, circuit input;
info_χ, information associated with χ, format varying by flag_χ.

Output.
$$\left\{\begin{array}{ll} \mathsf{Eval}_{\mathsf{normal}}(\chi, \mathsf{info}_\chi), & \text{if } \mathsf{flag}_\chi = \mathsf{normal}; \\ \mathsf{Eval}_{\mathsf{sim}}(\chi, \mathsf{info}_\chi), & \text{if } \mathsf{flag}_\chi = \mathsf{sim}. \quad \text{(Figure 7)} \end{array}\right.$$

$\mathsf{Expand}_{d,\mathsf{normal}}[\mathsf{pk}_{d+1}](\chi, \mathsf{info}_\chi)$

Hardwired. pk_{d+1}, public key for level $(d+1)$.

Input. $\chi \in \{0,1\}^d$, proper prefix of circuit input;
$\mathsf{info}_\chi = (C, \{k_{i,j}\}_{d \le i < D, 1 \le j \le B}, s_\chi)$:
C, circuit being obfuscated;
$k_{i,j}$, keys of H for level $d, \ldots, D-1$;
s_χ, seed of G_{sr} associated with χ.

Output. Computed as follows.

$s_{\chi\|0} \| r_{\chi\|0} \| s_{\chi\|1} \| r_{\chi\|1} \leftarrow G_{sr}(s_\chi)$
for $\eta = 0, 1$:
$\mathsf{flag}_{\chi\|\eta} \leftarrow \mathsf{normal}$
$\mathsf{info}_{\chi\|\eta} \leftarrow (C, \{k_{i,j}\}_{d+1 \le i < D, 1 \le j \le B}, s_{\chi\|\eta})$
$\mathsf{ct}_{\chi\|\eta} \leftarrow \mathsf{Enc}(\mathsf{pk}_{d+1}, \mathsf{flag}_{\chi\|\eta}, \chi\|\eta, \mathsf{info}_{\chi\|\eta})$
$\mathsf{otp}_\chi \leftarrow H(k_{d,1}, \chi\|0^{D-d}) \| \cdots \| H(k_{d,B}, \chi\|0^{D-d})$
output $v_\chi \leftarrow (\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}) \oplus \mathsf{otp}_\chi$

$\mathsf{Eval}_{\mathsf{normal}}(\chi, \mathsf{info}_\chi)$

Input. $\chi \in \{0,1\}^D$, circuit input;
$\mathsf{info}_\chi = (C, s_\chi)$:
C, circuit being obfuscated;
s_χ, unused.

Output. $C(\chi)$, computed by evaluating a universal circuit at (C, χ).

Figure 5. The circuits Expand_d and Eval in Construction 1 (branches for correctness).

Ingredients of Construction 1. Let

- D be the input length of the circuit C to be obfuscated;
- S the circuit size of C;
- L the block length, a parameter to be determined later;
- B the number of blocks, a parameter to be determined later;
- $H : \{0,1\}^\lambda \times \{0,1\}^D \to \{0,1\}^L$ the PRF of Pr$\mathcal{O}$M;
- $G_{sr} : \{0,1\}^\lambda \to \{0,1\}^{4\lambda}$ the PRG for encryption randomness;
- $G_v : \{0,1\}^\lambda \to \{0,1\}^L$ the PRG for decryption result simulation;
- $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ an FE scheme whose Enc uses λ-bit uniform randomness.

We construct an obfuscation scheme in the Pr$\mathcal{O}$ model for H:

Construction 1 (obfuscation)**.** $\mathsf{Obf}^{\mathcal{O}}(1^\lambda, C)$ does the following.

1. It sets up $(D+1)$ FE instances:
$$(\mathsf{pk}_D, \mathsf{sk}_D) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \mathsf{Eval}),$$
$$(\mathsf{pk}_d, \mathsf{sk}_d) \xleftarrow{\$} \mathsf{Gen}(1^\lambda, \mathsf{Expand}_d[\mathsf{pk}_{d+1}]) \quad \text{for } d = D-1, \dots, 0,$$
where Expand_d and Eval are defined in Figures 5 and 7.
2. It samples keys of H and obtains their handles:
$$k_{i,j} \xleftarrow{\$} \{0,1\}^\lambda, \quad h_{i,j} \leftarrow \mathcal{O}(\mathsf{hGen}, k_{i,j}) \qquad \text{for } 0 \le i < D,\ 1 \le j \le B.$$
3. It samples the seed and the encryption randomness for the root ciphertext, sets its flag and information, and computes ct_ε:
$$s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda, \qquad r_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda,$$
$$\mathsf{flag}_\varepsilon \leftarrow \mathsf{normal}, \qquad \mathsf{info}_\varepsilon \leftarrow (C, \{k_{i,j}\}_{0\le i<D, 1\le j\le B}, s_\chi),$$
$$\mathsf{ct}_\varepsilon \leftarrow \mathsf{Enc}(\mathsf{pk}_0, \mathsf{flag}_\varepsilon, \varepsilon, \mathsf{info}_\varepsilon; r_\varepsilon).$$
4. It outputs $\widehat{C}^\bullet[\mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\le d\le D}, \{h_{i,j}\}_{0\le i<D, 1\le j\le B}]$, defined in Figure 6, as an obfuscation of C.

Correctness. $\widehat{C}^{\mathcal{O}}$ in Figure 6 follows the tree structure in Figure 4. Correctness readily follows by inspecting the branches of Expand_d and Eval in Figure 5 and noting
$$H(k_{d,1}, \chi_d \| 0^{D-d}) \| \cdots \| H(k_{d,B}, \chi_d \| 0^{D-d})$$
$$= \mathcal{O}(\mathsf{hEval}, h_{d,1}, \chi_d \| 0^{D-d}) \| \cdots \| \mathcal{O}(\mathsf{hEval}, h_{d,B}, \chi_d \| 0^{D-d}).$$

Remarks. In our construction, the domain/codomain of the hash function are dependent on (the size and the input length of) the circuit being obfuscated. Formally, our obfuscator works in different Pr$\mathcal{O}$ models (relative to different PRFs) for circuits of different sizes and input lengths. However, it is simple to tweak the construction as well as the security proof so that the obfuscator works in a single Pr$\mathcal{O}$ model for $H : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}$ for all circuits. This is similar to how $\{0,1\}^* \to \{0,1\}$ implements the ROM for every domain/codomain by domain separation.

$\widehat{C}^{\mathcal{O}}[\mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0\le d\le D}, \{h_{i,j}\}_{0\le i<D, 1\le j\le B}](x)$

Hardwired. ct_ε, root ciphertext;
sk_d, secret keys;
$h_{i,j}$, handles of Pr$\mathcal{O}$M.

Input. $x \in \{0,1\}^D$, circuit input.

Output. Computed as follows.

for $d = 0, \ldots, D-1$:

$\chi_d \leftarrow x_{\le d}$

$v_{\chi_d} \leftarrow \mathsf{Dec}(\mathsf{sk}_d, \mathsf{ct}_{\chi_d})$

$\mathsf{otp}_{\chi_d} \leftarrow \mathcal{O}(\mathsf{hEval}, h_{d,1}, \chi_d \| 0^{D-d}) \| \cdots \| \mathcal{O}(\mathsf{hEval}, h_{d,B}, \chi_d \| 0^{D-d})$

$\mathsf{ct}_{\chi_d\|0} \| \mathsf{ct}_{\chi_d\|1} \leftarrow v_{\chi_d} \oplus \mathsf{otp}_{\chi_d}$

output $\mathsf{Dec}(\mathsf{sk}_D, \mathsf{ct}_x)$

Figure 6. The circuit $\widehat{C}^\bullet$ in Construction 1.

7 Security Proof of Ideal Obfuscation in the Pr$\mathcal{O}$ Model

Theorem 1 (¶). *Assuming PRF security of H, PRG security of G_{sr}, G_v, adaptive security (Definition 3) of* (Gen, Enc, Dec)*, and appropriate choice of L, B (Section 7.4), then Construction 1 is an ideal obfuscation (Definition 7) in the* Pr$\mathcal{O}$ *model (Definition 5) for H.*

We specify the simulator in Section 7.1 and prove Theorem 1 in Section 7.2.

Branches for Proof and Hybrid Template. The simulator for Construction 1 and the proof of Theorem 1 use the branches of Expand_d and Eval defined in Figure 7.

The hybrids as well as the simulator follow a common template shown in Figure 8, and we define a hybrid by specifying the placeholders in the hybrid. The three phases of the interaction are as follows:

- $\mathcal{S}_1$ (pre-obfuscation Pr$\mathcal{O}$M). In this phase, $\mathcal{S}_1$ efficiently implements the Pr$\mathcal{O}$M using lazy sampling.
- $\mathcal{S}_2$ (creating the obfuscation). In this phase, $\mathcal{S}_2$ generates FE keys, samples ("special") Pr$\mathcal{O}$M handles and keys used in the obfuscation, generates the root ciphertext ct_ε, and outputs the obfuscation. The handles are distinct, but the keys are not necessarily distinct (to facilitate application of PRF security), and not all handles correspond to a key. The placeholders specify which handles have a corresponding key and how ct_ε is generated.
- $\mathcal{S}_3$ (post-obfuscation Pr$\mathcal{O}$M). There are multiple cases, depending on whether the query is related to the "special" handles and keys:
  - For $(\mathsf{hGen}, k_{i,j})$, the output is $h_{i,j}$, as it should be. Not all $k_{i,j}$'s are considered in all hybrids (specified by the placeholders). Intuitively, this case can happen only with negligible probability, yet this very fact is proved using the hybrids and the branch is gradually removed as the proof proceeds.

– For $(\mathsf{hEval}, h_{i,j}, \chi\|0^{D-i})$, its output is supposed to unmask $\mathsf{Dec}(\mathsf{sk}_{|\chi|}, \mathsf{ct}_\chi)$ and specified by the placeholders.

– For $(\mathsf{hEval}, h_{i,j}, \cdot)$, the output is unrelated to obfuscation and specified by the placeholders.

– For the other queries (excluded from $(\mathsf{hGen}, k_{i,j})$ or unrelated to "special" handles and keys), it is the same as $\mathcal{S}_1$.

$\mathsf{Expand}_{d,\mathrm{hyb}}[\mathsf{pk}_{d+1}](\chi, \mathsf{info}_\chi)$

Hardwired. pk_{d+1}, public key for level $(d+1)$.

Input. $\chi \in \{0,1\}^d$, proper prefix of circuit input;
$\mathsf{info}_\chi = (C, \{k_{i,j}\}_{d<i<D, 1\le j\le B}, s_\chi, \beta,$
$\{\sigma_{\chi,j}\}_{1\le j<\beta}, w_\chi, \{k_{d,j}\}_{\beta<j\le B})$:
C, circuit being obfuscated;
$k_{i,j}$, keys of H for level $d+1, \ldots, D-1$;
s_χ, seed of G_{sr} associated with χ;
β, hybrid index;
$\sigma_{\chi,j}$, seeds of G_v associated with χ (gradually introduced);
w_χ, hardwired block of decryption result;
$k_{d,j}$, keys of H for level d (gradually removed).

Output. Computed as follows (difference from $\mathsf{Expand}_{d,\mathrm{normal}}$).

$s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|1}\|r_{\chi\|1} \leftarrow G_{sr}(s_\chi)$
for $\eta = 0, 1$:
$\mathsf{flag}_{\chi\|\eta} \leftarrow \mathsf{normal}$
$\mathsf{info}_{\chi\|\eta} \leftarrow (C, \{k_{i,j}\}_{d+1\le i<D, 1\le j\le B}, s_{\chi\|\eta})$
$\mathsf{ct}_{\chi\|\eta} \leftarrow \mathsf{Enc}(\mathsf{pk}_{d+1}, \mathsf{flag}_{\chi\|\eta}, \chi\|\eta, \mathsf{info}_{\chi\|\eta})$
output $v_\chi \leftarrow$ $G_v(\sigma_{\chi,1})\|\cdots\|G_v(\sigma_{\chi,\beta-1})\|w_\chi$
$\|\left([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{\beta+1} \oplus H(k_{d,\beta+1}, \chi\|0^{D-d})\right)\|\cdots$
$\|\left([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_B \oplus H(k_{d,B}, \chi\|0^{D-d})\right)$

$\mathsf{Expand}_{d,\mathrm{sim}}(\chi, \mathsf{info}_\chi)$

Input. $\chi \in \{0,1\}^d$, proper prefix of circuit input;
$\mathsf{info}_\chi = \{\sigma_{\chi,j}\}_{1\le j\le B}$, seeds of G_v associated with χ.

Output. $v_\chi \leftarrow G_v(\sigma_{\chi,1})\|\cdots\|G_v(\sigma_{\chi,B})$.

$\mathsf{Eval}_{\mathrm{sim}}(\chi, \mathsf{info}_\chi)$

Input. $\chi \in \{0,1\}^D$, circuit input;
$\mathsf{info}_\chi = y_\chi$, hardwired circuit output at χ.

Output. y_χ.

Figure 7. The circuits Expand_d and Eval in Construction 1 (branches for security proof).

Hybrid Template with Placeholders

Shared State:

$\mathcal{T}_{\text{other}}$, set of (k, h) pairs for hMap, initially $\varnothing$, with

$\mathsf{Keys}(\mathcal{T}_{\text{other}}) \stackrel{\text{def}}{=} \{\, k \mid \exists h \text{ such that } (k,h) \in \mathcal{T}_{\text{other}} \,\}$,

$\mathsf{Handles}(\mathcal{T}_{\text{other}}) \stackrel{\text{def}}{=} \{\, h \mid \exists k \text{ such that } (k,h) \in \mathcal{T}_{\text{other}} \,\}$;

C, circuit being obfuscated, available in $\mathcal{S}_2$ and $\mathcal{S}_3$;

$\{h_{i,j}\}_{0 \le i < D, 1 \le j \le B}$, handles of Pr$\mathcal{O}$M in obfuscation, initially $\perp$;

$\{\mathsf{pk}_d, \mathsf{sk}_d\}_{0 \le d \le D}$, public and secret keys, initially $\perp$;

$\{k_{i,j}\}$, keys of H in obfuscation, initially $\perp$;

$\{F_{i,j}\}, F_\sigma, F_r, F_s$, random functions (lazily sampled) for non-programmed portion of $\mathcal{O}(\mathsf{hEval}, h_{i,j}, \star)$, and $\sigma_{\chi,j}$, r_χ, s_χ;

$\mathsf{flag}_\chi, \mathsf{info}_\chi, r_\chi$, components of $\mathsf{ct}_\chi = \mathsf{Enc}(\mathsf{pk}_{|\chi|}, \mathsf{flag}_\chi, \chi, \mathsf{info}_\chi; r_\chi)$, available in $\mathcal{S}_2$ and $\mathcal{S}_3$.

$\mathcal{S}_1(\mathsf{hGen}, k)$:

if $\nexists h$ such that $(k,h) \in \mathcal{T}_{\text{other}}$:

$h \xleftarrow{\$} \{0,1\}^\lambda \setminus (\mathsf{Handles}(\mathcal{T}_{\text{other}}) \cup \{h_{i,j}\})$

$\mathcal{T}_{\text{other}} \leftarrow \mathcal{T}_{\text{other}} \cup \{(k,h)\}$

output the unique h such that $(k,h) \in \mathcal{T}_{\text{other}}$

$\mathcal{S}_1(\mathsf{hEval}, h, t)$:

if $\nexists k$ such that $(k,h) \in \mathcal{T}_{\text{other}}$:

$k \xleftarrow{\$} \{0,1\}^\lambda \setminus \mathsf{Keys}(\mathcal{T}_{\text{other}})$

$\mathcal{T}_{\text{other}} \leftarrow \mathcal{T}_{\text{other}} \cup \{(k,h)\}$

output $H(k,t)$ for the unique k such that $(k,h) \in \mathcal{T}_{\text{other}}$

$\mathcal{S}_2$:

generate $\{\mathsf{pk}_d, \mathsf{sk}_d\}_{0 \le d \le D}$ as specified in Construction 1

sample uniformly random distinct $h_{i,j}$ from $\{0,1\}^\lambda \setminus \mathsf{Handles}(\mathcal{T}_{\text{other}})$

$\{k_{i,j}\} \xleftarrow{\$} \{0,1\}^\lambda$

output $\widehat{C}^\bullet[\mathsf{ct}_\varepsilon, \{\mathsf{sk}_d\}_{0 \le d \le D}, \{h_{i,j}\}_{0 \le i < D, 1 \le j \le B}]$

$\mathcal{S}_3(\mathsf{hGen}, k)$:

if $k = k_{i,j}$ for "$k \stackrel{?}{=} k_{i,j}$", range of (i,j) being tested:

output $h_{i,j}$ for the smallest such (i,j)

else: same as $\mathcal{S}_1(\mathsf{hGen}, k)$

$\mathcal{S}_3(\mathsf{hEval}, h, t)$:

if $h = h_{i,j}$:

if $t = \chi \| 0^{D-i}$ for $\chi \in \{0,1\}^i$:

"$h_{i,j} : \chi$", response to $\mathcal{S}_3(\mathsf{hEval}, h_{i,j}, t = \chi \| 0^{D-i})$

else:

"$h_{i,j} : t$", response to $\mathcal{S}_3(\mathsf{hEval}, h_{i,j}, t \ne \chi \| 0^{D-i})$

else: same as $\mathcal{S}_1(\mathsf{hEval}, h, t)$

Figure 8. The hybrid template for the security proof of Construction 1.

7.1 Simulator

The simulator is specified in Table 1:

- ct_χ is in *simulation* mode and uses *truly random* r_χ for Enc;
- *no* $h_{i,j}$ has a corresponding PRF key; and
- ct_χ for $\chi \neq \varepsilon$ is *not computed* by $\mathsf{Expand}_{|\chi|-1}$, but *programmed* into the Pr$\mathcal{O}$M responses to $h_{|\chi|-1,j}$'s.

Although the template in Figure 8 has C (the circuit being obfuscated) as part of its share state, the template itself does not use C. The simulator only uses evaluations of C in $\mathcal{S}_3$ (to generate ct_χ for $|\chi| = D$ on demand), hence adheres to the required syntax of a simulator in Definition 7.

Table 1. Specification of the simulator (see Figure 8).

$\{k_{i,j}\}$	non-existent
$\{F_{i,j}\}$	$\{0,1\}^D \to \{0,1\}^L$ for $0 \le i < D$, $1 \le j \le B$
F_σ	$\{0,1\}^{<D} \times \{1,\ldots,B\} \to \{0,1\}^\lambda$
F_r	$\{0,1\}^{\le D} \to \{0,1\}^\lambda$
F_s	non-existent
flag_χ	sim
info_χ	$\begin{cases} \{F_\sigma(\chi,j)\}_{1\le j\le B}, & \text{if } \lvert\chi\rvert < D; \\ C(\chi), & \text{if } \lvert\chi\rvert = D. \end{cases}$
r_χ	$F_r(\chi)$
state $\mathcal{S}_3$	in simulator
$k \stackrel{?}{=} k_{i,j}$	non-existent
$h_{i,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]_j$
$h_{i,j} : t$	$F_{i,j}(t)$

7.2 Hybrids over Levels

To prove Theorem 1, we consider $\mathsf{Hyb}_{\delta,\star}$[11] for $0 \le \delta \le D$ specified in Table 2. The main hybrids are $\mathsf{Hyb}_{\delta,\$\$}$'s:

- ct_χ for $|\chi| < \delta$ is in *simulation* mode and uses *truly random* r_χ for Enc;
- ct_χ for $|\chi| = \delta$ is in *normal* mode and uses *truly random* r_χ;
- ct_χ for $|\chi| > \delta$ is in *normal* mode and uses *pseudorandom* r_χ expanded from $s_{\chi_{\le\delta}}$;
- $h_{i,j}$ for $i < \delta$ *does not have* a corresponding PRF key;
- $h_{i,j}$ for $i \ge \delta$ *has* a corresponding PRF key;

[11] The mnemonic is the form of s, r at level δ — in $\mathsf{Hyb}_{\delta,\$\$}$ they are truly random, and in $\mathsf{Hyb}_{\delta,G(\$)}$ they are PRG image.

- ct_χ for $0 < |\chi| \le \delta$ is *not computed* by $\mathsf{Expand}_{|\chi|-1}$, but *programmed* into the Pr$\mathcal{O}$M responses to $h_{|\chi|-1,j}$'s; and
- ct_χ for $|\chi| > \delta$ is *computed* by $\mathsf{Expand}_{|\chi|-1}$, *not programmed* into the Pr$\mathcal{O}$M.

The helper hybrids are $\mathsf{Hyb}_{\delta,s}$'s. Their only difference from the main hybrids is that (s,r) expansion starts at level $(\delta-1)$ instead of δ, i.e., ct_χ for $|\chi| \ge \delta$ is in *normal* mode and uses *pseudorandom* r_χ expanded from $s_{\chi_{\le\delta-1}}$.

The following lemmas hold for $\mathsf{Hyb}_{\delta,\star}$'s:

Lemma 2 (¶). *Let* $\mathsf{Hyb}_{\mathrm{real}}$ *be the real experiment (implicit in the minuend in Definition 7), then* $\mathsf{Hyb}_{\mathrm{real}} \approx_{\mathrm{s}} \mathsf{Hyb}_{0,\$\$}$.

Lemma 3 (¶). $\mathsf{Hyb}_{\delta,\$\$} \approx \mathsf{Hyb}_{\delta+1,G(\$)}$ *for all* $0 \le \delta < D$.

Lemma 4 (¶). $\mathsf{Hyb}_{\delta,G(\$)} \approx \mathsf{Hyb}_{\delta,\$\$}$ *for all* $1 \le \delta \le D$.

Lemma 5 (¶). *Let* $\mathsf{Hyb}_{\mathrm{sim}}$ *be the simulation experiment (implicit in the subtrahend in Definition 7), then* $\mathsf{Hyb}_{D,\$\$} \approx \mathsf{Hyb}_{\mathrm{sim}}$.

Table 2. Specification of $\mathsf{Hyb}_{\delta,\star}$ for $0 \le \delta \le D$ (see Figure 8).

$\{k_{i,j}\}$	$\delta \le i < D$, $1 \le j \le B$	
$\{F_{i,j}\}$	$\{0,1\}^D \to \{0,1\}^L$ for $0 \le i < \delta$, $1 \le j \le B$	
F_σ	$\{0,1\}^{<\delta} \times \{1,\dots,B\} \to \{0,1\}^\lambda$	
F_r	$\{0,1\}^{\le\delta} \to \{0,1\}^\lambda$	$\{0,1\}^{\le\delta-1} \to \{0,1\}^\lambda$
F_s	$\{0,1\}^{\delta} \to \{0,1\}^\lambda$	$\{0,1\}^{\delta-1} \to \{0,1\}^\lambda$
▸ $\|\chi\| < \delta$:		
flag_χ	sim	
info_χ	$\{F_\sigma(\chi,j)\}_{1\le j\le B}$	
r_χ	$F_r(\chi)$	
▸ $\|\chi\| = \delta$:		
r_χ	$F_r(\chi)$	expand from
s_χ	$F_s(\chi)$	$s_{\chi_{\le\delta-1}} = F_s(\chi_{\le\delta-1})$
▸ $\|\chi\| \ge \delta$:		
flag_χ	normal	
info_χ	$(C, \{k_{i,j}\}_{\|\chi\|\le i<D, 1\le j\le B}, s_\chi)$	
$\overset{(s,r)}{\text{expansion}}$	$s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|1}\|r_{\chi\|1} = G_{sr}(s_\chi)$	
state $\uparrow$ $\mathcal{S}_3$ $\downarrow$	in $\mathsf{Hyb}_{\delta,\$\$}$	in $\mathsf{Hyb}_{\delta,G(\$)}$
$k \overset{?}{=} k_{i,j}$	$\delta \le i < D$, $1 \le j \le B$ (checks for all existent $k_{i,j}$'s)	
▸ $i < \delta$:		
$h_{i,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_j$	
$h_{i,j} : t$	$F_{i,j}(t)$	
▸ $i \ge \delta$:		
$h_{i,j} : \chi$	$H(k_{i,j}, \chi\|0^{D-i})$	
$h_{i,j} : t$	$H(k_{i,j}, t)$	

Proof (Theorem 1). It follows from a standard hybrid argument over

$$\mathsf{Hyb}_{\mathrm{real}} \overset{2}{\approx}_{\mathrm{s}} \mathsf{Hyb}_{0,\$\$} \overset{3}{\approx} \mathsf{Hyb}_{1,G(\$)} \overset{4}{\approx} \mathsf{Hyb}_{1,\$\$} \overset{3}{\approx} \cdots \overset{4}{\approx} \mathsf{Hyb}_{D,\$\$} \overset{5}{\approx} \mathsf{Hyb}_{\mathrm{sim}},$$

where the number over each "$\approx$" references the lemma used. □

Lemmas 2, 4, and 5 are straightforward and we prove them below. We present the proof of Lemma 3 in Section 7.3.

Proof (Lemma 2). Starting from $\mathsf{Hyb}_{\mathrm{real}}$, the following modifications are made to reach $\mathsf{Hyb}_{0,\$\$}$:

1. Change $h_{i,j}$'s from being uniformly random to being uniformly random and distinct over $\{0,1\}^\lambda \setminus \mathsf{Handles}(\mathcal{T}_{\mathrm{other}})$.
2. Change $k_{i,j}$'s from being uniformly random and distinct over $\{0,1\}^\lambda \setminus \mathsf{Keys}(\mathcal{T}_{\mathrm{other}})$ to being uniformly random. ($k_{i,j}$'s are still excluded in the sampling of k's in $\mathcal{S}_3$.)
3. Stop excluding $k_{i,j}$'s in the sampling of k's in $\mathcal{S}_3$.

$\mathsf{Hyb}_{\mathrm{real}} \approx_{\mathrm{s}} \mathsf{Hyb}_{0,\$\$}$ follows from a standard birthday bound argument. □

Proof (Lemma 4). In $\mathsf{Hyb}_{\delta,G(\$)}$, all PRG seed s_χ for $|\chi| = \delta - 1$ has been removed from ct_χ and it is only used to obtain

$$s_{\chi\|0} \| r_{\chi\|0} \| s_{\chi\|1} \| r_{\chi\|1} = G_{sr}(s_\chi).$$

In $\mathsf{Hyb}_{\delta,\$\$}$, the left-hand side is replaced by true randomness. The two hybrids are otherwise identical, and their indistinguishability follows from the PRG security of G_{sr}.

The reduction is a hybrid argument over all s_χ for $\chi \in \{0,1\}^{\delta-1}$ such that at least one of $s_{\chi\|0}$, $r_{\chi\|0}$, $s_{\chi\|1}$, and $r_{\chi\|1}$ is used (to create $\mathsf{ct}_{\chi\|0}$ and $\mathsf{ct}_{\chi\|1}$) to respond to $\mathcal{S}_3(\mathsf{hEval}, h_{\delta-1,j}, \star)$. Since there are only polynomially queries from an efficient adversary, the reduction only incurs a polynomial loss of security. Hereafter, the same trick implicitly applies to the reduction implied by any other proof of this paper. □

Proof (Lemma 5). The only difference between $\mathsf{Hyb}_{D,\$\$}$ (Table 2 with $\delta = D$) and $\mathsf{Hyb}_{\mathrm{sim}}$ (Table 1) is the plaintext encrypted under ct_χ for $|\chi| = D$:

$$\begin{aligned} (\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) &= (\mathsf{normal}, \chi, (C, s_\chi)) && \text{in } \mathsf{Hyb}_{D,\$\$}, \\ (\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) &= (\mathsf{sim}, \chi, C(\chi)) && \text{in } \mathsf{Hyb}_{\mathrm{sim}}. \end{aligned}$$

In both hybrids, ct_χ's for $|\chi| = D$ are all encrypted using true randomness (that is not used elsewhere) under the public key pk_D. Since the secret key sk_D is for the function Eval and

$$\begin{aligned} \mathsf{Eval}(\mathsf{normal}, \chi, (C, s_\chi)) &= \mathsf{Eval}_{\mathrm{normal}}(\chi, (C, s_\chi)) \\ &= C(\chi) \\ &= \mathsf{Eval}_{\mathrm{sim}}(\chi, C(\chi)) = \mathsf{Eval}(\mathsf{sim}, \chi, C(\chi)), \end{aligned}$$

$\mathsf{Hyb}_{D,\$\$} \approx \mathsf{Hyb}_{\mathrm{sim}}$ reduces to the adaptive security of 1-key FE. □

Table 3. Specification of $\mathsf{Hyb}_{\delta,\beta,\star}$ for $0 \le \delta < D$, $1 \le \beta \le B$ (see Figure 8).

$\{k_{i,j}\}$	$\substack{i=\delta:\,\beta\le j\le B\\ \delta<i<D:\,1\le j\le B}$		$\substack{i=\delta:\,\beta+1\le j\le B\\ \delta<i<D:\quad 1\le j\le B}$		
$\{F_{i,j}\}$	$\substack{\{0,1\}^D\\ \to\{0,1\}^L}$ for $\substack{0\le i<\delta:\,1\le j\le B\\ i=\delta:\,1\le j<\beta}$		$\substack{0\le i<\delta:\,1\le j\le B\\ i=\delta:\,1\le j<\beta+1}$		
F_σ	$\substack{(\{0,1\}^{<\delta}\times\{1,\ldots,B\})\\ \cup(\{0,1\}^{\delta}\times\{1,\ldots,\beta-1\})}$ $\to \{0,1\}^\lambda$			$\{0,1\}^\lambda \leftarrow$ $\substack{(\{0,1\}^{<\delta}\times\{1,\ldots,B\})\\ \cup(\{0,1\}^{\delta}\times\{1,\ldots,\beta\})}$	
F_r	$\{0,1\}^{\le\delta} \to \{0,1\}^\lambda$				
F_s	$\{0,1\}^{\delta} \to \{0,1\}^\lambda$				
▸ $\lvert\chi\rvert < \delta$:					
flag_χ	sim				
info_χ	$\{F_\sigma(\chi,j)\}_{1\le j\le B}$				
r_χ	$F_r(\chi)$				
▸ $\lvert\chi\rvert = \delta$:					
r_χ	$F_r(\chi)$				
s_χ	$F_s(\chi)$				
$\boxed{w_\chi}$	$[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta \oplus H(k_{\delta,\beta}, \chi\|0^{D-\delta})$		$\substack{[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta\\ \oplus F_{\delta,\beta}(\chi\|0^{D-\delta})}$	$F_{\delta,\beta}(\chi\|0^{D-\delta})$	$G_v(F_\sigma(\chi,\beta))$
flag_χ	hyb				
info_χ	$(C, \{k_{i,j}\}_{\delta<i<D,1\le j\le B}, s_\chi, \beta,$ $\{F_\sigma(\chi,j)\}_{1\le j<\beta}, \boxed{w_\chi}, \{k_{\delta,j}\}_{\beta<j\le B})$				
▸ $\lvert\chi\rvert > \delta$:					
flag_χ	normal				
info_χ	$(C, \{k_{i,j}\}_{\lvert\chi\rvert\le i<D,1\le j\le B}, s_\chi)$				
$\substack{(s,r)\\ \text{expansion}}$	$s_{\chi\|0}\|r_{\chi\|0}\|s_{\chi\|1}\|r_{\chi\|1} = G_{sr}(s_\chi)$				

state ↑ $\mathcal{S}_3$ ↓	in $\mathsf{Hyb}_{\delta,\beta,1}$	in $\mathsf{Hyb}_{\delta,\beta,2}$	in $\mathsf{Hyb}_{\delta,\beta,3}$	in $\mathsf{Hyb}_{\delta,\beta,4}$	in $\mathsf{Hyb}_{\delta,\beta,5}$
$k \stackrel{?}{=} k_{i,j}$	$\substack{\delta<i<D:\,1\le j\le B\\ i=\delta:\,\beta\le j\le B}$	$\substack{\delta<i<D:\quad 1\le j\le B\\ i=\delta:\,\beta+1\le j\le B}$			
▸ $i < \delta$:					
$h_{i,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_j$				
$h_{i,j} : t$	$F_{i,j}(t)$				
▸ $i = \delta$ and $j < \beta$:					
$h_{\delta,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_j$				
$h_{\delta,j} : t$	$F_{\delta,j}(t)$				
▸ $i = \delta$ and $j = \beta$:					
$h_{\delta,\beta} : \chi$	$H(k_{\delta,\beta}, \chi\|0^{D-\delta})$		$F_{\delta,\beta}(\chi\|0^{D-\delta})$	$\substack{F_{\delta,\beta}(\chi\|0^{D-\delta})\\ \oplus[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta}$	$\substack{G_v(F_\sigma(\chi,\beta))\\ \oplus[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta}$
$h_{\delta,\beta} : t$	$H(k_{\delta,\beta}, t)$		$F_{\delta,\beta}(t)$	$F_{\delta,\beta}(t)$	$F_{\delta,\beta}(t)$
▸ $i = \delta$ and $j > \beta$:					
$h_{\delta,j} : \chi$	$H(k_{\delta,j}, \chi\|0^{D-\delta})$				
$h_{\delta,j} : t$	$H(k_{\delta,j}, t)$				
▸ $i > \delta$:					
$h_{i,j} : \chi$	$H(k_{i,j}, \chi\|0^{D-i})$				
$h_{i,j} : t$	$H(k_{i,j}, t)$				

7.3 Hybrids over Blocks at Each Level

To prove Lemma 3, we consider $\mathsf{Hyb}_{\delta,\beta,\star}$ for $0 \le \delta < D$, $1 \le \beta \le B$ specified in Table 3. In these hybrids, ct_χ's at level $|\chi| \neq \delta$ and responses to $\mathcal{S}_3(\mathsf{hEval}, h_{i,j}, \star)$ at level $i \neq \delta$ remain the same as in $\mathsf{Hyb}_{\delta,\$\$}$. We focus on the changes at level δ.

Recall that the decryption result is the one-time-padded child ciphertexts, which (at each level, and in particular, level δ) are split into B blocks, the one-time pad of each block corresponding to a handle $h_{\delta,j}$. In $\mathsf{Hyb}_{\delta,\beta,1}$, the first $(\beta-1)$ blocks have been switched to pseudorandom, and those blocks of the child ciphertexts, at level $(\delta+1)$, are hardwired into the responses to $\mathcal{S}_3(\mathsf{hEval}, h_{\delta,j}, \star)$ so that correctness is maintained. The β^{th} block is hardwired into the level-δ ciphertext, which is set to the block of the child ciphertexts padded using $k_{\delta,\beta}$. The last $(B-\beta)$ blocks are computed in the same way as in $\mathsf{Hyb}_{\delta,\$\$}$, from χ, s_χ, and $k_{i,j}$'s for $\delta < i < D$ and $1 \le j \le B$. The handles $h_{\delta,j}$ for $1 \le j < \beta$ do not have a corresponding PRF key, whereas $h_{\delta,j}$ for $\beta \le j \le B$ do.

Moving from $\mathsf{Hyb}_{\delta,\beta,1}$ to $\mathsf{Hyb}_{\delta,\beta+1,1}$:

- $\mathsf{Hyb}_{\delta,\beta,2}$ no longer checks $k \stackrel{?}{=} k_{\delta,\beta}$ in $\mathcal{S}_3(\mathsf{hGen}, k)$.
- $\mathsf{Hyb}_{\delta,\beta,3}$ replaces $H(k_{\delta,\beta}, \star)$, thus $\mathcal{S}_3(\mathsf{hEval}, h_{\delta,\beta}, \star)$, by random function $F_{\delta,\beta}$.
- $\mathsf{Hyb}_{\delta,\beta,4}$ makes the β^{th} block of the decryption result random and *programs* the β^{th} block of the child ciphertexts into $\mathcal{S}_3(\mathsf{hEval}, h_{\delta,\beta}, \star)$.
- $\mathsf{Hyb}_{\delta,\beta,5}$ makes the β^{th} block of the decryption result *pseudorandom*.
- $\mathsf{Hyb}_{\delta,\beta+1,1}$ collects the PRG seed for the β^{th} block and recycles the hardwiring space for the $(\beta+1)^{\text{st}}$ block.

The following lemmas hold for $\mathsf{Hyb}_{\delta,\beta,\star}$'s:

Lemma 6 (¶). $\mathsf{Hyb}_{\delta,\$\$} \approx \mathsf{Hyb}_{\delta,1,1}$ *for all* $0 \le \delta < D$.

Lemma 7 (¶). *For all* $0 \le \delta < D$ *and* $1 \le \beta \le B$,

$$\mathsf{Hyb}_{\delta,\beta,1} \approx \mathsf{Hyb}_{\delta,\beta,2} \approx \mathsf{Hyb}_{\delta,\beta,3} \equiv \mathsf{Hyb}_{\delta,\beta,4} \approx \mathsf{Hyb}_{\delta,\beta,5}.$$

Lemma 8 (¶). $\mathsf{Hyb}_{\delta,\beta,5} \approx \mathsf{Hyb}_{\delta,\beta+1,1}$ *for all* $0 \le \delta < D$ *and* $1 \le \beta < B$.

Lemma 9 (¶). $\mathsf{Hyb}_{\delta,B,5} \approx \mathsf{Hyb}_{\delta+1,G(\$)}$ *for all* $0 \le \delta < D$.

Proof (Lemma 3). It follows from a standard hybrid argument over

$$\begin{aligned}\mathsf{Hyb}_{\delta,\$\$} \overset{6}{\approx} \mathsf{Hyb}_{\delta,1,1} \overset{7}{\approx} \mathsf{Hyb}_{\delta,1,5} \overset{8}{\approx} \mathsf{Hyb}_{\delta,2,1} \\ \overset{7}{\approx} \mathsf{Hyb}_{\delta,2,5} \overset{8}{\approx} \cdots \overset{7}{\approx} \mathsf{Hyb}_{\delta,B,5} \overset{9}{\approx} \mathsf{Hyb}_{\delta+1,G(\$)},\end{aligned}$$

where the number over or under each "≈" references the lemma used. □

It remains to prove Lemmas 6, 7, 8, and 9.

Proof (Lemma 6). The only difference between $\mathsf{Hyb}_{\delta,\$\$}$ (Table 2) and $\mathsf{Hyb}_{\delta,1,1}$ (Table 3) is the plaintext encrypted under ct_χ for $|\chi| = \delta$:

$$(\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) = (\mathsf{normal}, \chi, (C, \{k_{i,j}\}_{\delta \le i < D, 1 \le j \le B}, s_\chi)) \qquad \text{in } \mathsf{Hyb}_{\delta,\$\$},$$

$$(\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) = (\mathsf{hyb}, \chi, (C, \{k_{i,j}\}_{\delta<i<D, 1\le j\le B}, s_\chi, \overbrace{1}^{\beta},$$
$$\underbrace{\varnothing}_{\{\sigma_{\chi,j}\}_{1\le j<1}}, \underbrace{[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_1 \oplus H(k_{\delta,1}, \chi\|0^{D-\delta})}_{w_\chi}, \{k_{\delta,j}\}_{1<j\le B})) \quad \text{in } \mathsf{Hyb}_{\delta,1,1}.$$

In both hybrids, ct_χ's for $|\chi| = \delta$ are encrypted using true randomness (that is not used elsewhere) under pk_δ, and sk_δ is for the function Expand_δ. As

$$\begin{aligned}
&\mathsf{Expand}_\delta(\mathsf{normal}, \chi, (C, \{k_{i,j}\}_{\delta\le i<D, 1\le j\le B}, s_\chi)) \\
&= (\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}) \oplus (H(k_{\delta,1}, \chi\|0^{D-\delta})\|\cdots\|H(k_{\delta,B}, \chi\|0^{D-\delta})) \\
&= ([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_1 \oplus H(k_{\delta,1}, \chi\|0^{D-\delta})) \\
&\quad \|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_2 \oplus H(k_{\delta,2}, \chi\|0^{D-\delta}))\|\cdots \\
&\quad \|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_B \oplus H(k_{\delta,B}, \chi\|0^{D-\delta})) \\
&= \mathsf{Expand}_\delta(\mathsf{hyb}, \chi, (C, \{k_{i,j}\}_{\delta<i<D, 1\le j\le B}, s_\chi, 1, \\
&\qquad \varnothing, [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_1 \oplus H(k_{\delta,1}, \chi\|0^{D-\delta}), \{k_{\delta,j}\}_{1<j\le B})),
\end{aligned}$$

$\mathsf{Hyb}_{\delta,\$\$} \approx \mathsf{Hyb}_{\delta,1,1}$ reduces to the adaptive security of 1-key FE. □

Proof (Lemma 7). For $\mathsf{Hyb}_{\delta,\beta,1} \approx \mathsf{Hyb}_{\delta,\beta,2}$, the two are identical until ("bad event") the adversary queries $\mathcal{S}_3(\mathsf{hGen}, k_{\delta,\beta})$. Prior to the bad event, the only interaction of the adversary with $k_{\delta,\beta}$ amounts to querying its evaluations at various points. For appropriate choice of B (Section 7.4), the bad event can happen only with negligible probability due to the PRF security of H. Therefore, the two hybrids are indistinguishable.

$\mathsf{Hyb}_{\delta,\beta,2} \approx \mathsf{Hyb}_{\delta,\beta,3}$ reduces to the PRF security of H, because in $\mathsf{Hyb}_{\delta,\beta,2}$, the PRF key $k_{\delta,\beta}$ is only used for evaluating at various points (thanks to removing $k \stackrel{?}{=} k_{\delta,\beta}$ in the previous step).

$\mathsf{Hyb}_{\delta,\beta,3} \equiv \mathsf{Hyb}_{\delta,\beta,4}$ is the perfect secrecy of one-time pad.

$\mathsf{Hyb}_{\delta,\beta,4} \approx \mathsf{Hyb}_{\delta,\beta,5}$ reduces to the PRG security of G_v. □

Proof (Lemma 8). The only difference between $\mathsf{Hyb}_{\delta,\beta,5}$ and $\mathsf{Hyb}_{\delta,\beta+1,1}$ is the plaintext encrypted under ct_χ for $|\chi| = \delta$:

$$\begin{aligned}
(\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) = (\mathsf{hyb}, \chi, (C, \{k_{i,j}\}_{\delta<i<D, 1\le j\le B}, s_\chi, \beta, \\
\{F_\sigma(\chi, j)\}_{1\le j<\beta}, \\
G_v(F_\sigma(\chi, \beta)), \\
\{k_{\delta,j}\}_{\beta<j\le B})) \quad \text{in } \mathsf{Hyb}_{\delta,\beta,5}, \\
(\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi) = (\mathsf{hyb}, \chi, (C, \{k_{i,j}\}_{\delta<i<D, 1\le j\le B}, s_\chi, \beta+1, \\
\{F_\sigma(\chi, j)\}_{1\le j<\beta+1}, \\
[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta \oplus H(k_{\delta,\beta+1}, \chi\|0^{D-1}), \\
\{k_{\delta,j}\}_{\beta+1<j\le B})) \quad \text{in } \mathsf{Hyb}_{\delta,\beta+1,1}.
\end{aligned}$$

In both hybrids, ct_χ's for $|\chi| = \delta$ are encrypted using true randomness (that is not used elsewhere) under pk_δ, and sk_δ is for the function Expand_δ. It suffices to verify

$$\begin{aligned}
&\mathsf{Expand}_\delta(\mathsf{hyb}, \chi, (C, \{k_{i,j}\}_{\delta<i<D, 1\le j\le B}, s_\chi, \beta, \\
&\qquad \{F_\sigma(\chi, j)\}_{1\le j<\beta}, G_v(F_\sigma(\chi, \beta)), \{k_{\delta,j}\}_{\beta<j\le B}))
\end{aligned}$$

$$
\begin{aligned}
&= G_v(F_\sigma(\chi,1))\|\cdots\|G_v(F_\sigma(\chi,\beta-1))\\
&\quad \|G_v(F_\sigma(\chi,\beta))\|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{\beta+1}\oplus H(k_{\delta,\beta+1},\chi\|0^{D-\delta}))\\
&\quad \|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{\beta+2}\oplus H(k_{\delta,\beta+2},\chi\|0^{D-\delta}))\|\cdots\\
&\quad \|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{B}\oplus H(k_{\delta,B},\chi\|0^{D-\delta}))\\
&= \mathsf{Expand}_\delta(\mathsf{hyb},\chi,(C,\{k_{i,j}\}_{\delta<i<D,1\le j\le B},s_\chi,\beta+1,\\
&\qquad\qquad \{F_\sigma(\chi,j)\}_{1\le j<\beta+1},\\
&\qquad\qquad [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{\beta+1}\oplus H(k_{\delta,\beta+1},\chi\|0^{D-\delta}),\\
&\qquad\qquad \{k_{\delta,j}\}_{\beta+1<j\le B})),
\end{aligned}
$$

and $\mathsf{Hyb}_{\delta,\beta,5}\approx\mathsf{Hyb}_{\delta,\beta+1,1}$ reduces to the adaptive security of 1-key FE. □

Proof (Lemma 9). The only difference between $\mathsf{Hyb}_{\delta,B,5}$ (Table 3) and $\mathsf{Hyb}_{\delta+1,G(\$)}$ (Table 2) is the plaintext encrypted under ct_χ for $|\chi|=\delta$:

$$
\begin{aligned}
(\mathsf{flag}_\chi,\chi,\mathsf{info}_\chi) &= (\mathsf{hyb},\chi,(C,\{k_{i,j}\}_{\delta<i<D,1\le j\le B},s_\chi,B,\\
&\qquad \{F_\sigma(\chi,j)\}_{1\le j<B},\underbrace{G_v(F_\sigma(\chi,B))}_{w_\chi},\underbrace{\varnothing}_{\{k_{\delta,j}\}_{B<j\le B}})) \quad \text{in } \mathsf{Hyb}_{\delta,B,5},\\
(\mathsf{flag}_\chi,\chi,\mathsf{info}_\chi) &= (\mathsf{sim},\chi,\{F_\sigma(\chi,j)\}_{1\le j\le B}) \qquad\qquad \text{in } \mathsf{Hyb}_{\delta+1,G(\$)}.
\end{aligned}
$$

In both hybrids, ct_χ's for $|\chi|=\delta$ are encrypted using true randomness (that is not used elsewhere) under pk_δ, and sk_δ is for the function Expand_δ. It holds that

$$
\begin{aligned}
&\mathsf{Expand}_\delta(\mathsf{hyb},\chi,(C,\{k_{i,j}\}_{\delta<i<D,1\le j\le B},s_\chi,B,\\
&\qquad\qquad \{F_\sigma(\chi,j)\}_{1\le j<B},G_v(F_\sigma(\chi,B)),\varnothing))\\
&= G_v(F_\sigma(\chi,1))\|\cdots\|G_v(F_\sigma(\chi,B-1))\|G_v(F_\sigma(\chi,B))\\
&= \mathsf{Expand}_\delta(\mathsf{sim},\chi,\{F_\sigma(\chi,j)\}_{1\le j\le B}),
\end{aligned}
$$

so $\mathsf{Hyb}_{\delta,B,5}\approx\mathsf{Hyb}_{\delta+1,G(\$)}$ reduces to the adaptive security of 1-key FE. □

7.4 Choice of Parameters

We will set $L=B$. Let n,m be the length of plaintexts and circuits in 1-key FE. Expand_d and Eval (Figures 5 and 7) have

- a universal circuit for circuits up to size S,
- 2 copies of Enc of 1-key FE,
- B copies of H,
- B copies of G_v, and
- other components of size $\mathrm{poly}(\lambda)$ or subsumed by the above.

Suppose the encryption circuit is of size at most $(n^{2-2\varepsilon}+m^{1-\varepsilon})\lambda^{e_1}$ for some constant $e_1>0$ and $0<\varepsilon<1/2$.[12] We have (below, $|\cdot|$ is the bit length of everything)

$$
n=|\mathsf{flag}|+|\chi|+|C|+DB|k|+|s|+|\beta|+B|\sigma|+|w|\le SDB\lambda^{e_2}
$$

[12] This derivation assumes $\lambda\ge 2$ and $n,m\ge 1$. We also assume $1\le D,S,L,B\le 2^\lambda$.

for some constant $e_2 > 0$. For representing the circuit, we need

$$m = \Omega(S \log S) + 2(n^{2-2\varepsilon} + m^{1-\varepsilon})\lambda^{e_1} + BL\,\mathrm{poly}(\lambda)\log L + \mathrm{poly}(\lambda),$$

for which we require

$$m \geq (S^2 D^2 B^2 + m^{1-\varepsilon})\lambda^{e_3}$$

for a certain constant $e_3 > 0$. For sufficiently long one-time pads, we also need

$$2(n^{2-2\varepsilon} + m^{1-\varepsilon})\lambda^{e_1} \leq LB = B^2.$$

To satisfy these constraints, it suffices to set

$$L = B = 2m^{(1-\varepsilon)/2}\lambda^{e_1+e_2}, \quad m = (5S^2 D^2 \lambda^{2e_1+2e_2+e_3})^{1/\varepsilon}.$$

Acknowledgments. Aayush Jain was supported by gifts from CyLab of CMU and Google. Huijia Lin and Ji Luo were supported by NSF CNS-1936825 (CAREER), CNS-2026774, a JP Morgan AI Research Award, a Cisco Research Award, and a Simons Collaboration on the Theory of Algorithmic Fairness. Daniel Wichs was supported by NSF CNS-1750795, CNS-2055510, and the JP Morgan Faculty Research Award. The authors thank the anonymous reviewers for their valuable feedback.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015.

[ACFQ22] Prabhanjan Ananth, Kai-Min Chung, Xiong Fan, and Luowen Qian. Collusion-resistant functional encryption for RAMs. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 160–194. Springer, Heidelberg, December 2022.

[AIK+21] Shweta Agrawal, Yuval Ishai, Eyal Kushilevitz, Varun Narayanan, Manoj Prabhakaran, Vinod M. Prabhakaran, and Alon Rosen. Secure computation from one-way noisy communication, or: Anti-correlation via anti-concentration. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 124–154, Virtual Event, August 2021. Springer, Heidelberg.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. https://eprint.iacr.org/2015/730.

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 162–172. Springer, Heidelberg, December 2014.

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001.

[BBC+14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 26–51. Springer, Heidelberg, February 2014.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 520–537. Springer, Heidelberg, August 2010.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

[BCG+11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 722–739. Springer, Heidelberg, December 2011.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BIK+22] Saikrishna Badrinarayanan, Yuval Ishai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Refuting the dream XOR lemma via ideal obfuscation and resettable MPC. In Dana Dachman-Soled, editor, *ITC 2023*, volume 230 of *LIPIcs*, pages 10:1–10:21. Schloss Dagstuhl, 2022.

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. Can we access a database both locally and privately? In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 662–693. Springer, Heidelberg, November 2017.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 456–467. Springer, Heidelberg, March 2015.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015.

[CRRV17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-ciphertext secure fully homomorphic encryption. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 213–240. Springer, Heidelberg, March 2017.

[DIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 519–535. Springer, Heidelberg, August 2013.

[DMMN11] Nico Döttling, Thilo Mie, Jörn Müller-Quade, and Tobias Nilges. Basing obfuscation on simple tamper-proof hardware assumptions. Cryptology ePrint Archive, Report 2011/675, 2011. https://eprint.iacr.org/2011/675.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th FOCS*, pages 102–115. IEEE Computer Society Press, October 2003.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013,*

Part II, volume 8043 of *LNCS*, pages 536–553. Springer, Heidelberg, August 2013.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 194–213. Springer, Heidelberg, February 2007.

[GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Heidelberg, October / November 2016.

[HHWW19] Ariel Hamlin, Justin Holmgren, Mor Weiss, and Daniel Wichs. On the plausibility of fully homomorphic encryption for RAMs. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 589–619. Springer, Heidelberg, August 2019.

[IKLS22] Yuval Ishai, Alexis Korb, Paul Lou, and Amit Sahai. Beyond the Csiszár-korner bound: Best-possible wiretap coding via obfuscation. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 573–602. Springer, Heidelberg, August 2022.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, March 2015.

[JLL23] Aayush Jain, Huijia Lin, and Ji Luo. On the optimal succinctness and efficiency of functional encryption and attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 479–510. Springer, Heidelberg, April 2023.

[JLLW23] Aayush Jain, Huijia Lin, Ji Luo, and Daniel Wichs. The pseudorandom oracle model and ideal obfuscation. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 233–262. Springer, Heidelberg, August 2023.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Heidelberg, May / June 2022.

[JP18] Marc Joye and Alain Passelègue. Function-revealing encryption - definitions and constructions. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 527–543. Springer, Heidelberg, September 2018.

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obfustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 603–648. Springer, Heidelberg, April / May 2018.

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019.

[LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Heidelberg, October / November 2016.

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In Barna Saha and Rocco A. Servedio, editors, *55th ACM STOC*, pages 595–608. ACM Press, June 2023.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 447–462. Springer, Heidelberg, March 2016.

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

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

[NFR+17] Kartik Nayak, Christopher W. Fletcher, Ling Ren, Nishanth Chandran, Satya V. Lokam, Elaine Shi, and Vipul Goyal. HOP: Hardware makes obfuscation practical. In *NDSS 2017*. The Internet Society, February / March 2017.

[Nis22] Ryo Nishimaki. Personal communication, 2022.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, April 1979.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.

[Zha22] Mark Zhandry. To label, or not to label (in generic groups). In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 66–96. Springer, Heidelberg, August 2022.