The Pseudorandom Oracle Model and Ideal Obfuscation

Aayush Jain[1] Huijia Lin[2] Ji Luo[2] Daniel Wichs[3]

[1] Carnegie Mellon University
aayushjain1728@gmail.com

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

[3] Northeastern University and NTT Research
wichs@ccs.neu.edu

September 2022

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 a 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 the oracle maps it to a public handle h. Given the handle h and some input x, the oracle will recover the PRF key k and evaluate the PRF on x. 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 gives a heuristic justification for the following: (1) most natural security goals implied by ideal obfuscation are achievable in the real world; (2) we can construct obfuscation from FE with 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.

Contents

1 Introduction

Hash Functions vs. Random Oracles. Hash functions are one of the most important cryptographic building blocks and are ubiquitous in both theoretical and practical cryptosystem design. The basic security property of hash functions is collision resistance, and this property already suffices for many applications. However, there is a widespread belief that good hash functions can satisfy a much wider range of cryptographic security properties beyond collision resistance. This belief is captured via 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 design cryptosystems and give formal proofs of security. We then take a heuristic leap of faith that such cryptosystems remain secure when we replace the random oracle by a real, well-designed hash function (like SHA-3). While the second step is heuristic and does not have a formal justification, it captures the intuition that an adversary cannot meaningfully do anything with a well-designed hash function beyond 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, we can design 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]. But outside of such specially crafted and contrived 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-life cryptosystem.

Indistinguishability vs. Ideal Obfuscation. A scenario analogous to the one above also plays out in the upper reaches of cryptomania 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 cryptographic security properties beyond $i\mathcal{O}$. In fact, the evidence of this is even stronger in the case of obfuscation, due to the fact that any $i\mathcal{O}$ is guaranteed to be the "best possible obfuscator" [GR07] — if some obfuscator is capable of satisfying some security property, then $i\mathcal{O}$ must satisfy this property as well.[1] 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.[2] Analogously to the ROM, we can construct cryptosystems and formally prove their security in the ideal obfuscation model, which is extraordinarily powerful and gives very simple constructions. Then, we can then make a heuristic leap of faith that such cryptosystems remains secure when we replace the ideal

[1] Modulo adding some padding of appropriate size to the programs being obfuscated.

[2] Ideal obfuscation is similar to the notion of 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, which required 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.

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.

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". The main motivating question for this work is

Can we formalize the connection between
ideal hash functions and ideal obfuscation?

As a starting point, we might ask whether it is possible to construct 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] answered the above question in the negative and showed that it is impossible to construct ideal obfuscation in the random oracle model.

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 but standard-model primitive, which can in turn be based on well-studied polynomial hardness assumptions. We believe that this result formalizes the following intuition:

Heuristically, assuming we have ideal obfuscation is
tightly connected to heuristically assuming 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.

Next, we first discuss the Pr$\mathcal{O}$ model in detail and justify why we believe it is a reasonable idealized model for hash functions, similar to the basic ROM. We then discuss how we interpret our construction of ideal obfuscation in the Pr$\mathcal{O}$ model, in light of the fact that ideal obfuscation is impossible in the plain model. Lastly, we give a technical overview of our construction.

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 real pseudorandom function (PRF) family H_k. The ideal functionality has two interfaces. The first interface is used to *initialize* a hash function by providing as input a PRF key k: the ideal functionality maps the PRF key k to a random *handle* h and outputs

it. The second interface is used to *evaluate* the hash function by providing a handle h and an input x: the ideal functionality finds the corresponding PRF key k for the handle 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 get oracle access to the hash function H_k by querying the oracle with the handle h. By pseudorandomness of H_k, this is indistinguishable from a truly random oracle. However, the Pr$\mathcal{O}$ 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 evaluating it in a non-black-box way without making any calls to the oracle. 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 guarantees — 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 of these descriptions for the same hash function and different users can use different descriptions. 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 the hash function H_k then all security guarantees are lost! In effect, this means that 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 captures the idea that, although our overall cryptosystem relies on non-black-box use of the hash function, we can only rely on Pr$\mathcal{O}$ security in hybrids where all non-black-box use of the hash function is removed.

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 contain the keys k_i inside some functional encryption (FE) ciphertext. The evaluator will get the FE ciphertext as well as the corresponding handles h_i, which will suffice to evaluate 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 analyze security, we will need a careful sequence of hybrids where we replace Pr$\mathcal{O}$ outputs for the handle h_i by random values in hybrids where the PRF key k_i is removed from the FE ciphertext.

Interpreting Our Result on Ideal Obfuscation. Ideal obfuscation cannot be realized by any real obfuscation scheme, similarly to the fact that a random oracle cannot be realized by any real hash function. But, also similar to random oracles, ideal obfuscation provides a formal model in which we can design and analyze cryptosystems. Later, we can instantiate them using a real-life obfuscator based on the intuition that a good

obfuscator is sufficient to achieve what ideal obfuscation achieves in most reasonable scenarios. This ideal obfuscation heuristic is powerful.

- First, 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: doubly efficient PIR [BIPW17], fully Homomorphic encryption for RAM [HHWW19], realizing oblivious transfer using binary erasure channels [AIK+21], input-hiding obfuscation for evasive functions [BBC+14], virtual-gray-box obfuscation [BC10], public-coin differing input obfuscation [IPS15], succinct obfuscation and functional encryption for unbounded-input Turing Machines [IPS15], extractable witness encryption and attribute-based encryption for Turing machine and RAM [GKP+13], counterexamples to the XOR lemma [BIK+22], wiretap-channel coding [IKLS22] etc. This list will surely continue to grow.

- Second, 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 an obfuscated program that decrypts ciphertexts of a (CCA secure) public key encryption and then computes the function f on the encrypted input. In contrast, $i\mathcal{O}$ applications typically involve more sophisticated techniques (e.g., the punctured program technique [SW14]), in order to overcome the weak security of $i\mathcal{O}$, producing cumbersome constructions with complex proofs of security. The idea obfuscation heuristic gives strong evidence that this is unnecessary in real life.

- Third, for many real-life security goals, such as, obfuscating machine learning models, protecting software patches, creating cripple-ware where parts of the functionality are redacted, etc, $i\mathcal{O}$ security is insufficient. In these specific natural contexts, the 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 real-life hash function, provides an alternative option. The main advantage of our obfuscator is that it is based on polynomial hardness assumptions, as opposed to subexponential hardness.

Alternate Interpretations: Hardware Tokens, 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$\mathcal{O}$ for providing black-box accesses to H_k) and the obfuscated program that relies on k. There are several prior works that showed how to construct obfuscation using hardware tokens [DMMN11,BCG+11,NFR+17]. However, in all cases, the hardware token is more significantly more complex than just implementing a PRF. Therefore, we believe our work also provides an interesting new take on how to construct obfuscation using extremely simple hardware tokens.

Alternately, 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 obfus-

cated program that implements H_k.[3] In the literature, there are several bootstrapping theorems that transform obfuscation for weak classes of functions to obfuscation for general functions (e.g., [GGH+13,App14,CLTV15]). In these works, the weak classes are typically weaker from a complexity theoretic perspective, such as, belonging to NC_1 or TC_0, but are expressive enough to hardcode an arbitrary circuit in the function description (e.g., f is able to verify that a ciphertext is the correct output ciphertext obtained by homomorphically evaluate a circuit C on some input ciphertexts, and if so decrypts that ciphertext). In comparison, our bootstrapping theorem starts with obfuscation of a single PRF family $\{H_k\}$, which is much simpler.

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

Assume we have a real-world cryptosystem that uses some real hash function, say SHA-3 with a *public seed* k. Some users/components of the cryptosystem will only make black-box calls to the hash function $SHA3(k,\cdot)$, but do not rely on the code otherwise, while other users/components of the cryptosystem may use the code of the hash function $SHA3(k,\cdot)$ in a non-black-box way. In the usual ROM, we model the former use-case by replacing all calls to $SHA3(k,\cdot)$ with oracle calls to a truly random public oracle, but do not have any way of capturing the latter use-case. In the Pr$\mathcal{O}$ model, we can set $H_k(\cdot) = SHA3(k,\cdot)$[4] and replace all black-box calls to $SHA3(k,\cdot)$ with oracle calls to $H_k(\cdot)$[5]. Additionally, if the original cryptosystem also used the hash function $SHA3(k,\cdot)$ in a non-black-box way, the Pr$\mathcal{O}$ models allows parties knowing k to make non-black-box use of H_k, while ensuring that it is consistent with all of the black-box calls to the hash function, which were replaced by oracle calls.

The above describes why it is reasonable to take any real-world cryptosystem that relies on black-box and non-black-box use of a hash function such as SHA-3 and model it in the Pr$\mathcal{O}$. We are also be interested in the reverse direction. Like RO, the Pr$\mathcal{O}$ model articulates an explicit design paradigm: For a crypto problem Π, to design a good scheme (or protocol) P for Π:

(1) Find a formal definition for Π in the model of computation in which all parties (including the adversary) share the pseudorandom oracle $\mathcal{O}$.

(2) Devise an efficient scheme P for Π in this Pr$\mathcal{O}$ model.

(3) Prove that P satisfies the definition for Π in the Pr$\mathcal{O}$ model.

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

[4] Assume that $SHA3(k,\cdot)$ is a PRF with key k, which is a very mild real-world hash function assumption.

[5] Formally, we model this by letting the pseudorandom oracle publish some handle h corresponding to the key k and we replace all black-box calls to $SHA3(k,\cdot)$ with calls to an oracle, where the call provides the handle h and the oracle translates it back to k and evaluates $SHA3(k,\cdot)$.

(4) Instantiate the pseudorandom oracle $\mathcal{O}$ using a real hash function.

One possible instantiation using $SHA3$ is setting $H_k(x) = SHA3(k, x)$, replace the handle h with $(SHA3, k)$ and every evaluation call to Pr$\mathcal{O}$ with an evaluation of $SHA3(k, \cdot)$.

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 last step (Step (4)) is heuristic in nature. There are known schemes/protocols secure in the ROM, but insecure when instantiated with any real hash functions, e.g., [Bar01,GK03]. These counterexamples extends to the Pr$\mathcal{O}$ model. In addition, our construction of ideal obfuscation is an example that 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 that are proven 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 real-world 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.

Another dimension regards the benefits of having a formal ideal model over ad hoc heuristics in individual use-cases. The ROM was motivated and guided by heuristic uses of hash functions that preceded it, such as, the Fiat-Shamir transformation [FS87]. However, the benefits of having an explicit ideal model go beyond providing partial justification to these use-cases. 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 in the ROM [BCCT13], in construction of simulation-secure functional encryption [DIJ+13], and chosen-ciphertext secure fully homomorphic encryption [CRRV17]. It is well-motivated to formalize a variant of the ROM able to capture some non-black-box uses of hash functions. However, such efforts quickly meets 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 (as in recursive proof composition) have been deemed less satisfactory than heuristics justified in the ROMs. The Pr$\mathcal{O}$ model side-steps the contradiction–the oracle evaluates pseudorandom functions $H_k(\cdot)$ that has efficient code representation. However, as discussed above, 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.

Overall, we believe that the Pr$\mathcal{O}$ model is a natural and more flexible variant of the ROM that allows us to formally reason about cryptosystems that make non-black box use of hash functions, while being "morally analogous" to the ROM. At first, the idea of an ideal model that captures 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 satisfying answer to this. We can only rely in 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 also assume that the adversary only has black-box access.

A Technical Subtlety. Before proceeding to the technical body, we explain a technical subtlety related to the formalization of Pr$\mathcal{O}$ and its instantiation. There are two options for modeling the public handle h in Pr$\mathcal{O}$: 1) model h as a special symbol that cannot be operated on (i.e., cannot be viewed as a string and computed on) and can only be used as an argument in evaluation calls to Pr$\mathcal{O}$; 2) model h as a (sufficiently long) randomly chosen string. The former only permits designing schemes that use h to make black-box calls of the hash function, whereas the latter allows schemes that may use h in an arbitrary way. In the former modeling, we can instantiate $H_k(\cdot) = SHA3(k, \cdot)$ and $h = (SHA3, k)$ as described above, and all uses of h for black-box calls are replaced with evaluation $SHA(k, x)$. In the latter model, the same instantiation may run into unnatural corner cases: For instance, the scheme may check whether h contains k as a substring. This is not the case in the Pr$\mathcal{O}$ model where h is a random string, but is the case when h is instantiated as $(SHA3, k)$. In this case, we need more care. Instead, we can heuristically instantiate this by setting $H_k = SHA3(SHA3(k), \cdot)$ and $h = (SHA3, SHA3(k))$, and replace all oracle calls using an input h with calls to $SHA3(h, \cdot)$. "Non-black-box" uses of h in the scheme becomes non-black-box use of $(SHA3, SHA3(k))$. The distinction between the two models are similar to that between Maurer's [Mau05] and Shoup's [Sho97] generic group model, which is recently studied in the work of [Zha22]. We view these two formalizations as morally the same. Our construction of ideal obfuscation only uses h for black-box evaluation calls and can be formalized and proven secure in either model. In the technical body, we use the second option, as it gives constructions more flexibility.

Organization. We overview our techniques in Section 2. After providing preliminaries in Section 3, we define the Pr$\mathcal{O}$ model in Section 4 and ideal obfuscation in Section 5. We present our construction of ideal obfuscation in Section 6 and prove its security in Section 7.

1.1 Related Works

We are aware of several prior works that have attempted to rely on random oracle security of a hash function while simultaneously making non-black-box use of the hash function. For example, the work of Valiant [Val08] constructs incrementally verifiable proofs of knowledge and the work of [DIJ+13] constructs simulation-based 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 specified ideal model in which we can state and prove results. On the other hand, the Pr$\mathcal{O}$ model only allows very careful non-black-box use of a 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 directly translate into results in the Pr$\mathcal{O}$ model.

2 Technical Overview

Now we describe the main ideas behind our construction. The starting idea comes from the insights made by [BV15,AJ15] and the follow-ups [LPST16,BNPW16,KNT18,KNTY19], which establish that $i\mathcal{O}$ can realized generically from a seemingly weaker primitive of

a subexponentially secure single-key functional encryption scheme (FE). These results additionally require the FE scheme to satisfy certain encryption efficiency guarantees. In the overview below, assume that the FE scheme satisfies adaptive indistinguishability security and compactness with linear input dependency.[6] Refer to Definition 1 for a definition of a functional encryption scheme.

FE-to-$i\mathcal{O}$ Transformation. The idea is a really natural one. In order to obfuscate circuit $C : \{0,1\}^D \to \{0,1\}$ we give out an FE ciphertext ct_ε encrypting the circuit C. We think of this ciphertext as being associated with the root of a complete binary tree of depth D. We also give out FE secret keys for each of the D levels in the tree, with 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 $\mathsf{ct}_{x||0}, \mathsf{ct}_{x||1}$ associated with the children, where each such child ciphertext carries the information about the circuit C. Finally, for the ciphertexts ct_x at the leaf level with $x \in \{0,1\}^D$, we give out FE secret keys that allow 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 $i \in [0, D]$, run $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{FE.Gen}(1^\lambda)$.
- Compute $\mathsf{ct}_\varepsilon \xleftarrow{\$} \mathsf{FE.Enc}(\mathsf{pk}_0, \mathsf{info}_\varepsilon)$, where $\mathsf{info}_\varepsilon = (C, \varepsilon, \star)$ where C is the circuit, and $\star$ indicates a slot that will contain some useful programming information used in the proof (such as PRF keys) that will be specified later as needed.
- Generate $\mathsf{sk}_{f_i} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{sk}_i, f_i)$ for $i \in [0, D]$. Here the function f_i for $i \in [0, D-1]$ under normal functioning, takes as input a ciphertext ct_χ for $\chi \in \{0,1\}^i$, encrypting $\mathsf{info}_\chi = (C, \chi, \star)$ under pk_i and produces two ciphertexts $(\mathsf{ct}_{\chi\|0}, \mathsf{ct}_{\chi\|1})$ encrypting $\mathsf{info}_{\chi\|b} = (C, \chi\|b, \star)$ for $b \in \{0,1\}$ respectively under pk_{i+1}. Finally, f_D takes as input $\mathsf{info}_x = (C, x \in \{0,1\}^D, \star)$ and outputs $C(x)$.

The output of the obfuscation is $\hat{C} = \{\mathsf{ct}_\varepsilon, \{\mathsf{sk}_{f_i}\}_{i\in[0,D]}\}$. In order to evaluate the circuit $\hat{C}$ on input $x \in \{0,1\}^D$, one computes ct_x at level D, and then decrypts it using sk_{f_D} to compute $C(x)$. The process of computing ct_x is inductive and proceeds like a binary tree traversal. Let $x^{\le i}$ be the prefix of x of length i. We start by decrypting ct_ε using sk_{f_0} to compute $(\mathsf{ct}_0\|\mathsf{ct}_1)$. For $i \in [D-1]$, we inductively decrypt $\mathsf{ct}_{x^{\le i}}$ using sk_{f_i} to derive $\mathsf{ct}_{x^{\le i}\|0}\|\mathsf{ct}_{x^{\le i}\|1}$. We then use $\mathsf{ct}_{x^{\le i+1}}$ to continue this way along the tree.

The scheme described above satisfies *polynomial slowdown* because of the compactness of FE scheme. Since at every level, ct_χ encrypts info_χ the running time of each function f_i is $O(|\mathsf{info}_\chi|\,\mathrm{poly}(\lambda))$ and thus each f_i is polynomial sized. Although, some bit of care is needed because we need to ensure that the size of the slots denoted by $\star$ used by any ciphertext don't 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, the adversary can produce at least 2^D intermediate ciphertexts, ct_χ for $\chi \in \{0,1\}^{\le D}$, each encrypting $\mathsf{info}_\chi = (C, \chi, \star)$ containing the circuit that is being obfuscated. If C_0, C_1 are the two equivalent circuits,

[6] An FE scheme is said to satisfy compactness with linear input dependency, if the size of the encryption circuit is $|x|\,\mathrm{poly}(\lambda)$, where x is the message that is encrypted, and λ is the security parameter. This is independent of the function for which the keys are issued.

we would like to show that $\hat{C}_0$ is indistinguishable to $\hat{C}_1$. The adversary is given ct_ε, and $(\mathsf{sk}_{f_0}, \ldots, \mathsf{sk}_{f_D})$. The adversary can actually compute ct_χ for any $\chi \in \{0,1\}^{\leq D}$ of his choice, and can do so internally without the reduction knowing at all. Thus, the reduction needs a strategy to switch every ct_χ from containing C_0 to C_1 that the adversary can touch upon without being able to guess which ciphertexts ct_χ will be touched by the adversary. Therefore to achieve this, the proof works input by input. For every $x \in \{0,1\}^D$, the reduction switches every ciphertext encountered in the path to x, and their neighbors to use C_1 instead of C_0. This will require the use of the slots $\star$, along with PRF puncturing and hardwiring intermediate ciphertexts (within the keys/ciphertext ct_ε) in the path in the slots containing $\star$. Thus, going over all $x \in \{0,1\}^D$ introduces at least 2^D hybrids which result in a subexponential security loss.

Why Does the Scheme Fail to Be an Ideal Obfuscation? In an ideal obfuscation scheme, we require that $\tilde{C}$ can be simulated by a *polynomial-time* simulator having access only to an oracle implementing circuit C. What this means is that the simulator has to come up with a short ciphertext ct_ε, that is powerful enough to generate ct_x for every $x \in \{0,1\}^D$ containing the information enough to evaluate $C(x)$, without having access to the circuit in the clear and only as an oracle. Assuming hard-to-learn functions exist, this is impossible as the simulator can only query the oracle of the circuit C a polynomial number of times. Indeed, this trivial argument shows that ideal obfuscation cannot exist, which is also implied by the the work of [BGI+01], which showed that even a more restricted version of 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], where there are several known applications that are known to be impossible in the standard model but can be constructed in the random oracle model (Such as SNARKs).

A Simplified Intuition Using Random Oracles. From the observation above, one of the limitations behind proving the above scheme secure is that once the reduction/simulator produces some ciphertext ct_ε, this fully specifies all the intermediate ciphertexts in the tree and all the outputs of the circuit. There is no other place to "program" any information.The random oracle might be very useful in solving this issue. Imagine, a world in which ct_ε is set so that, the result of the first decryption is $H(\varepsilon) \oplus [\mathsf{ct}_0 \| \mathsf{ct}_1]$ so that adversary has to query $H(\varepsilon)$ to unmask next layer ciphertexts. Similarly, assume that 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 was hypothetically possible, then, we might be able to come up with a simulation strategy. The point is that a random oracle gives us two powerful capabilities that enable simulation of this kind: *observability* and *programmability*. As the evaluator queries $H(\chi)$ for various values of χ, the simulator can keep track of which path the adversary is taking to evaluate the ciphertext. Secondly, due to programming, one can undetectably move to a setting where the ciphertext start decrypting to a random value otp_χ as opposed to $H(\chi) \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$. Simultaneously, we can program the random oracle to respond to $H(\chi)$ by answering $\mathsf{otp}_\chi \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$.

Once this happens, any ct_χ can only be accessed by querying the oracle. Then the hope is that since there are only polynomial queries an adversary can make, we need to simulate only a polynomial of ciphertexts. In particular the goal would be to replace ct_χ for $\chi \in \{0,1\}^{\leq D-1}$ by "dummy" ciphertexts independent of the circuit C, and ct_x for $x \in \{0,1\}^D$ using simulated ciphertexts generated using $C(x)$. This will yield an ideal

obfuscation scheme with polynomial security.

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

Using the Pr$\mathcal{O}$ Model. The flaw with the idea above is the premise itself. We assumed that the decryption of ct_χ is of the form $H(\chi) \oplus \cdots$, but this requires the FE scheme to evaluate the hash function H. This only makes sense if the functions f_i in the FE secret keys depend on the code of the hash function H, which means that we need H to be a real hash function and not a random oracle! So we have seemingly conflicting requirements, where we need the code of the hash function to define the scheme syntactically, but we also need to treat the hash function as an ideal oracle to take advantage of observability and programmability.

This is where our model comes in. We precisely show that the above approach can be realized in the Pr$\mathcal{O}$ model. Let us briefly recall the Pr$\mathcal{O}$ model (see Definition 5 for a detailed definition). In this model, there are two oracle algorithms hGen and hEval with the syntax below.

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

where $\mathcal{O}(\mathsf{hGen}, \star)$ maps a key k into a handle h, and $\mathcal{O}(\mathsf{hEval}, h, x)$ maps the handle h back into its unique key k and outputs $H(k, x)$ where $H(k, \cdot)$ is some specific function family. The map that turns handles into the key and vice-versa is implemented by a random permutation and can be simulated efficiently using "lazy sampling". Further, we require that H is a pseudorandom function — given a handle h for a randomly chosen k, the function $\mathcal{O}(\mathsf{hEval}, h, \star) = H(k, \star)$ is computationally indistinguishable to a random function (when k is absent from the view of the adversary).

This model provides the right abstraction needed to solve our problem. A random key k (corresponding to handle h) can be used inside the FE ciphertext to compute $H(k, \chi) \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$. At the same time, if we could remove k from the adversary's view, we can still program $\mathcal{O}(\mathsf{hEval}, h, \star)$. Of course, the difficulty is that if the the key k is inside the FE ciphertext then it is part of the adversary's view. Therefore, we need to come up with a careful strategy involving a sequence of hybrids where we only program the oracle $\mathcal{O}(\mathsf{hEval}, h, \star)$ in hybrids where the key k is not present. We describe how to do so below.

First Attempt. We now describe our first attempt at the scheme. The scheme will use a PRG $G : \{0,1\}^\lambda \to \{0,1\}^{4\lambda}$ which will be used derive the randomness used for encrypting intermediate ciphertexts. Let λ be the length of Pr$\mathcal{O}$ hash keys, and the randomness used to compute FE ciphertexts. To obfuscate a circuit C, one computes ct_ε which encrypts $(C, \varepsilon, k, s_\varepsilon)$, where k is the hash key for which h is a handle. Further s_ε is a randomness that will be used to derive randomness for lower level ciphertexts. The function key f_0 takes as input $(C, \varepsilon, k, s_\varepsilon)$ and outputs $H(k, \varepsilon) \oplus [\mathsf{ct}_0 \| \mathsf{ct}_1]$. Functions $f_1, \ldots, f_D$ are described analogously. The output of the obfuscation is $\hat{C} = (h, \mathsf{ct}_\varepsilon, \{\mathsf{sk}_{f_i}\}_{i \in [0, D-1]})$. We give an outline in Figure 1.

Evaluating such an obfuscated circuit is straightforward. The idea is that ct_ε encrypts a Pr$\mathcal{O}$ hash key k, that will be used to compute masks $H(k, \chi)$ of the lower layer ciphertexts ct_χ. The evaluator is also provided a corresponding handle h, which can be used to derive $H(k, \chi)$ by making oracle calls. Thus, continuing as before one can compute ct_x, which can be used with sk_{f_D} to derive $C(x)$.

Ideal Obfuscation - First Attempt

Input: Circuit C, **Computation:**

- Sample $\Pr\mathcal{O}$ handle and key pair (h, k).
- Sample $r_\varepsilon, s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
- For $i \in [0, D]$, sample $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{FE.Gen}(1^\lambda)$. Generate $\mathsf{sk}_{f_i} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{sk}_i, f_i)$ for functions f_i described below.
- Set $\mathsf{ct}_\varepsilon = \mathsf{FE.Enc}(\mathsf{pk}_0, \mathsf{info}_\varepsilon; r_\varepsilon)$ where $\mathsf{info}_\varepsilon = (\mathsf{normal}, C, \varepsilon, k, s_\varepsilon)$. Here, normal indicates normal mode of the ciphertext. In the simulation mode, it will be switched with sim.
- Output $\hat{C} = (h, \mathsf{ct}_\varepsilon, \mathsf{sk}_{f_0}, \ldots, \mathsf{sk}_{f_D})$.

Function f_i: In the normal mode, the functions f_i for $i \in [0, D-1]$ work as follows. Let $\chi \in \{0,1\}^i$. $f_i(\mathsf{info}_\chi = (\mathsf{normal}, C, \chi, k, s_\chi))$ computes the following:

- Expand $G(s_\chi) = (s_{\chi\|0} \| r_{\chi\|0} \| s_{\chi\|1} \|_{\chi\|1})$.
- For $b \in \{0,1\}$ compute $\mathsf{ct}_{\chi\|b} = \mathsf{FE.Enc}(\mathsf{pk}_{i+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$ where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, k, s_{\chi\|b}))$.
- Output $H(k, \chi) \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$.

Function $f_D(\mathsf{info}_x = (\mathsf{normal}, C, x, k, s_x)) = C(x)$.

Figure 1. First Attempt.

For the security proof, unfortunately, while the intuition is clear where we want to program $\mathcal{O}(\mathsf{hEval}, h, \star)$ to go to a simulation mode, this can't be done in the scheme. The point is that in Pr$\mathcal{O}$ model, $\mathcal{O}(\mathsf{hEval}, h, \star)$ can only be programmed if k is not given out to the adversary. On the other hand, in the scheme k appears inside all the intermediate ciphertexts ct_χ. Thus it is not clear how to simulate even the first ciphertext ct_ε as the key k appears in the future ciphertexts. We will have to remove k from all the intermediate ciphertexts at once to appeal to the programmability of $\mathcal{O}(\mathsf{hEval}, h, \star)$. It is not clear how to do this.

Second Attempt. To deal with this issue, we make a slight change. We will use different key/handle pairs (h_i, k_i) for every layer $i \in [0, D-1]$ as opposed to a single pair. The idea is that a ciphertext ct_χ for $\chi \in \{0,1\}^d$ in layer $d \in [0, D-1]$ will only contain $\{(h_i, k_i)\}_{i \in [d, D-1]}$. In particular, a ciphertext won't contain keys used in the previous layers. This will enable to break out of the issue described above. The scheme is described in Figure 2.

Our high-level simulation strategy is inductive with respect to the layers. Suppose we manage to simulate ciphertexts ct_χ where $\chi^{\leq d}$ for $d \in [0, D-2]$ so that they have no information about $(k_0, \ldots, k_d)$, we will use this to simulate ct_χ for $\chi \in \{0,1\}^{d+1}$. The idea is that k_d is missing from the view of the adversary within the first d layers. This will

Ideal Obfuscation - Second Attempt

Input: Circuit C, **Computation:**

- Sample Pr$\mathcal{O}$ handle and key pairs (h_i, k_i) for $i \in [0, D-1]$.
- Sample $r_\varepsilon, s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
- For $i \in [0, D]$, sample $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{FE.Gen}(1^\lambda)$. Generate $\mathsf{sk}_{f_i} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{sk}_i, f_i)$ for functions f_i described below.
- Set $\mathsf{ct}_\varepsilon = \mathsf{FE.Enc}(\mathsf{pk}_0, \mathsf{info}_\varepsilon; r_\varepsilon)$ where $\mathsf{info}_\varepsilon = (\mathsf{normal}, C, \varepsilon, \{k_i\}_{i \in [0, D-1]}, s_\varepsilon)$. Here, normal indicates normal mode of the ciphertext. In the simulation mode, it will be switched with sim.
- Output $\hat{C} = (\{h_i\}_{i \in [0, D-1]}, \mathsf{ct}_\varepsilon, \mathsf{sk}_{f_0}, \ldots, \mathsf{sk}_{f_D})$.

Function f_i: In the normal mode, the functions f_i for $i \in [0, D-1]$ work as follows. Let $\chi \in \{0,1\}^i$. $f_i(\mathsf{info}_\chi = (\mathsf{normal}, C, \chi, \{k_d\}_{d \in [i, D-1]}, s_\chi))$ computes the following:

- Expand $G(s_\chi) = (s_{\chi\|0} \| r_{\chi\|0} \| s_{\chi\|1} \|_{\chi\|1})$.
- For $b \in \{0,1\}$ compute $\mathsf{ct}_{\chi\|b} = \mathsf{FE.Enc}(\mathsf{pk}_{i+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$ where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, \{k_d\}_{d \in [i+1, D-1]}, s_{\chi\|b}))$.
- Output $H(k_i, \chi) \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$.

Function $f_D(\mathsf{info}_x = (\mathsf{normal}, C, x, s_x)) = C(x)$.

Figure 2. Second Attempt.

enable us to program and observe $\mathcal{O}(\mathsf{hEval}, h_d, \chi)$ and thus we will have an opportunity to replace ct_χ with a dummy ciphertext.

In order to demonstrate this strategy, we will start from the very top and show how the first and subsequent iterations look like. Our eventual goal is to generate all the ciphertexts so that they don't contain information about the circuit C. To achieve that we will first make sure that the ciphertexts don't contain information about the keys $\{k_i\}_{i\in[0,D-1]}$ and use that to program the ciphertexts to contain no information about the circuit C. At the last layer, we will use the oracle of C to simulate ct_x. Consider the ciphertext ct_ε which encrypts $\mathsf{info}_\varepsilon = (\mathsf{normal}, C, \varepsilon, k_0, s_\varepsilon)$, where the first component indicates the "mode", which is set to normal in real life. We take the following steps at the very first layer.

- We replace ct_ε from encrypting $\mathsf{info}_\varepsilon = (\mathsf{normal}, C, \varepsilon, k_0, s_\varepsilon)$ to encrypting $\mathsf{info}_\varepsilon = (\mathsf{sim}, H(k_0, \varepsilon) \oplus [\mathsf{ct}_0 \| \mathsf{ct}_1])$, thereby changing the mode to sim and hardwiring the output of the first decryption. The function f_0 is set so that in the simulation mode, it outputs the hardwired value within the ciphertext. This change is indistinguishable due to the security of FE.

- Now, we can move to simulating the response of $\mathcal{O}(\mathsf{hEval}, h_0, \varepsilon)$ by truly random string otp_ε. This change is indistinguishable because k_0 is hidden from adversary's view.

- Then, we start generating ct_ε by encrypting $(\mathsf{sim}, \mathsf{otp}_\varepsilon)$ and simultaneously simulating the response of $\mathcal{O}(\mathsf{hEval}, h_0, \varepsilon)$ by responding $\mathsf{otp}_\varepsilon \oplus [\mathsf{ct}_0 \| \mathsf{ct}_1]$.

- Now one can simulate ct_0 and ct_1 inductively.

In fact, this strategy can be generalized for any layer $i \in [0, D-1]$. The simulator only needs to compute any given ct_χ for $\chi \in \{0,1\}^{i+1}$, only when asked for $\mathcal{O}(\mathsf{hEval}, h_i, \eta)$ where η is length i prefix of χ. An adversary can only ask for polynomial such queries as it is polynomial time. In the final layer, when the adversary asks for $\mathcal{O}(\mathsf{hEval}, h_{D-1}, \chi)$ for $\chi \in \{0,1\}^{D-1}$, the challenger responds by answering $\mathsf{otp}_\chi \oplus [\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$ where $\mathsf{ct}_{\chi\|b}$ encrypts $(\mathsf{sim}, C(\chi\|b))$. The last key outputs the hardwired value $C(x)$ in the simulation mode.

While the overall intuition above is really clear, there is an important issue we overlooked. The issue is that ct_χ needs to hardwire strings of length $|\mathsf{ct}_{\chi\|0} \| \mathsf{ct}_{\chi\|1}|$. Thus, the generation time of ct_χ grows at least exponentially as D increases. To resolve this issue, we revisit the simulation strategy at the very first layer itself and then leverage the idea to the all the layers.

Fixing Simulation Efficiency. The observation above suggests that the problem with the simulation strategy is present in the very first step itself. The ciphertext ct_ε cannot hardwire information $\mathsf{otp}_\varepsilon \oplus [\mathsf{ct}_0 \| \mathsf{ct}_1]$. We will resolve this issue by utilizing a much a smaller slot for hardwiring.

To demonstrate that, let the length $[\mathsf{ct}_0 \| \mathsf{ct}_1]$ be denoted by L_{ct}. We also define a parameter B, which will be denote the number of blocks. We will set B and L so that $B \cdot L = L_{\mathsf{ct}}$. The idea then is that instead of sampling one key handle pair for the first layer (h_0, k_0), we will sample B pairs $\{(h_{0,j}, k_{0,j})\}_{j\in[B]}$. We will do this for each layer. For $i \in [0, D-1]$ we sample B pairs $\{(h_{i,j}, k_{i,j})\}_{i\in[0,D-1], j\in[B]}$. Each $\mathcal{O}(\mathsf{hEval}, h_{i,j}, \chi)$ is now required to produce outputs of length L. This will allow us to switch a real ciphertext

out for a dummy ciphertext via a sequence of hybrids, where in each hybrid we only hardwire one L-bit block. In particular, the hybrids look as follows:

- Remove the j'th key $k_{i,j}$ from the ciphertext and hardwire the j'th block of the output in the ciphertext.
- Replace the hard-coded block by a random value and program the Pr$\mathcal{O}$ to maintain consistency. Note that we can program the random oracle on handle $h_{i,j}$ since the key $k_{i,j}$ was removed.
- Replace the hard-coded block (and the corresponding output of the Pr$\mathcal{O}$) by a pseudorandom value generated using a small PRG seed σ_j.
- Include the PRG seed σ_j in the ciphertext and use it to generate the j'th output block, removing all hard-coding.

Overall this allows us to only hard-code one L bit block and B small PRG seeds in the ciphertext for a total input size of $O(L+B)$ while generating an output of size $O(L \cdot B)$. While these are our main ideas, working them out requires a bit of care. We now describe the

Ideal Obfuscation - Final Construction

Input: Circuit C, **Computation:**

- Sample Pr$\mathcal{O}$ handle and key pairs $(h_{i,j}, k_{i,j})$ for $i \in [0, D-1]$ and $j \in [B]$.
- Sample $r_\varepsilon, s_\varepsilon \xleftarrow{\$} \{0,1\}^\lambda$.
- For $i \in [0, D]$, sample $(\mathsf{pk}_i, \mathsf{sk}_i) \xleftarrow{\$} \mathsf{FE.Gen}(1^\lambda)$. Generate $\mathsf{sk}_{f_i} \xleftarrow{\$} \mathsf{FE.KeyGen}(\mathsf{sk}_i, f_i)$ for functions f_i described below.
- Set $\mathsf{ct}_\varepsilon = \mathsf{FE.Enc}(\mathsf{pk}_0, \mathsf{info}_\varepsilon; r_\varepsilon)$ where $\mathsf{info}_\varepsilon = (\mathsf{normal}, C, \varepsilon, \{k_{i,j}\}_{i\in[0,D-1], j\in[B]}, s_\varepsilon)$. Here, normal indicates normal mode of the ciphertext. In the simulation mode, it will be switched with sim.
- Output $\hat{C} = (\{h_{i,j}\}, \mathsf{ct}_\varepsilon, \mathsf{sk}_{f_0}, \ldots, \mathsf{sk}_{f_D})$.

Function f_i: In the normal mode, the functions f_i for $i \in [0, D-1]$ work as follows. Let $\chi \in \{0,1\}^i$. $f_i(\mathsf{info}_\chi = (\mathsf{normal}, C, \chi, \{k_{d,j}\}_{d\in[i,D-1], j\in[B]}, s_\chi))$ computes the following:

- Expand $G(s_\chi) = (s_{\chi\|0} \| r_{\chi\|0} \| s_{\chi\|1} \|_{\chi\|1})$.
- For $b \in \{0,1\}$ compute $\mathsf{ct}_{\chi\|b} = \mathsf{FE.Enc}(\mathsf{pk}_{i+1}, \mathsf{info}_{\chi\|b}; r_{\chi\|b})$ where $\mathsf{info}_{\chi\|b} = (\mathsf{normal}, C, \chi\|b, \{k_{d,j}\}_{d\in[i+1,D-1], j\in[B]}, s_{\chi\|b}))$.
- Output $\mathsf{otp}_\chi \oplus [ct_{\chi\|0} \| \mathsf{ct}_{\chi\|1}]$ where $\mathsf{otp}_\chi = H(k_{i,1}, \chi) \| \ldots \| H(k_{i,B}, \chi)$.

Function $f_D(\mathsf{info}_x = (\mathsf{normal}, C, x, s_x)) = C(x)$.

Figure 3. Final Construction.

construction in a bit more detail. Then, we describe how the proof will work at the top layer. The same idea can be extended to simulate all the layers of ciphertexts. In more detail, instead of computing ct_ε as an encryption of $(C, \varepsilon, \{k_i\}_{i\in[0,D-1]}, s_\varepsilon)$, we generate it as an encryption of $(\mathsf{normal}, C, \varepsilon, \{k_{i,j}\}_{i\in[0,D-1],j\in[B]}, s_\varepsilon)$. The result of the decryption of ct_ε with sk_{f_0} will now compute $\mathsf{otp}_\varepsilon \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]$ where $\mathsf{otp}_\varepsilon = H(k_{0,1}, \varepsilon)\| \ldots \|H(k_{0,B}, \varepsilon)$. The construction is outlined in Figure 3.

- First we remove the key $k_{0,1}$ from the ciphertext. To do this, the ciphertext ct_ε is "partially simulated" using a new mode hyb. We encrypt $(\mathsf{hyb}, C, \varepsilon, \{k_{0,j}\}_{j>1}, \{k_{i,j}\}_{i\in[1,D-1],j\in[B]}, s_\varepsilon, \alpha_1)$. Here α_1 is the first L sized block of $\mathsf{otp}_\varepsilon \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]$ where $\mathsf{otp}_\varepsilon = H(k_{0,1}, \varepsilon)\| \ldots \|H(k_{0,B}, \varepsilon)$. The function f_0 is designed so that consistency is ensured in the outputs. This change is indistinguishable due to the security of FE
- Now that $k_{0,1}$ is missing, we program $\mathcal{O}(\mathsf{hEval}, h_{0,1}, \varepsilon)$. We generate α_1 as $\mathsf{otp}_{\varepsilon,1} \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]_1$ where $\mathsf{otp}_{\varepsilon,1}$ is truly random. We simultaneously program $\mathcal{O}(\mathsf{hEval}, h_{0,1}, \varepsilon)$ to output $\mathsf{otp}_{\varepsilon,1}$ to maintain consistency. This change is undetectable due to the security of $\Pr\mathcal{O}$.
- Then, we set α_1 as a random sample $\mathsf{otp}_{\varepsilon,1}$ and program $\mathcal{O}(\mathsf{hEval}, h_{0,1}, \varepsilon)$ to output $\mathsf{otp}_{\varepsilon,1} \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]_1$. The change is identical in distribution.
- Then, we set $\mathsf{otp}_{\varepsilon,1} = G(\sigma_{\varepsilon,1})$ where G is an appropriately expanding PRG and $\sigma_{\varepsilon,1}$ is a λ bit string. The change is undetectable due to the security of PRG.
- The point of all this is that now one can encrypt $(\mathsf{hyb}, C, \varepsilon, \sigma_{\varepsilon,1}, \{k_{0,j}\}_{j>1}, \{k_{i,j}\}_{i\in[1,D-1],j\in[B]}, s_\varepsilon)$ where the function f_0 is designed so that it maintain the consistency of outputs.
- Going this way, and removing $\{k_{0,j}\}_{j\in[B]}$ one by one, we can now come up to a stage where all the keys of the top level are removed and ct_ε is computed by encrypting $(\mathsf{hyb}, C, \varepsilon, \{\sigma_{\varepsilon,j}\}_{j\in[B]}, \{k_{i,j}\}_{i\in[1,D-1],j\in[B]}, s_\varepsilon)$. At this point, we can simply move to a simulated mode by encrypting $(\mathsf{sim}, \varepsilon, \{\sigma_{\varepsilon,j}\}_{j\in[B]})$ and the function f_0 simply outputs $G(\sigma_{\varepsilon,1})\| \ldots \|G(\sigma_{\varepsilon,B})$. At this point, the oracle query $\mathcal{O}(\mathsf{hEval}, h_{\varepsilon,j}, \varepsilon)$ is programmed to output $G(\sigma_{\varepsilon,j}) \oplus [\mathsf{ct}_0\|\mathsf{ct}_1]_j$.

The basic idea described above can be carefully extended to work at all the layers, simulating a layer at a time. For any $\chi \in \{0,1\}^i$, we have already established that the only way to generate ct_χ is by querying the oracles $\mathcal{O}(\mathsf{hEval}, h_{i-1,j}, \star)$. Thus, the simulator can now program ct_χ. Further, an adversary can only make a polynomial of queries. Thus, we only need to generate a polynomial number of ciphertexts. Each time the simulator can essentially plays the same set of hybrids as above to replace each queried ciphertext to a simulated ciphertext analogous to one described above. This works for all the layers except the final layer D. In that case, we use $C(x)$ to simulate the ciphertext which can be done because the simulator has access to an oracle implementing C, and can simulate the last layer as well by making polynomial calls to C.

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 $(a, b) \le (c, d)$ means either $a = c$ and $b \le d$ or $a < c$, for integers a, b, c, d.

We write $x \| y$ for the concatenation of two strings x, y. The empty string is denoted by ε. Given a string x and a length $0 \le i \le |x|$, we let $x_{\le i}$ 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 $j \ge 1$.

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 $\ell_{\text{in}} \ell_{\text{out}} \operatorname{poly}(\lambda)$.

Functional Encryption. We base our obfuscation scheme on 1-key functional encryption, which is weaker than the standard notion:[7]

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

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

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.

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

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

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,JLL22] studying the transformations among functional encryption schemes with various security and efficiency guarantees. It is known [JLL22,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.[8] 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 an oracle, 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 $C^{\mathcal{O}}(x)$.

[8] 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 define the pseudorandom oracle model Pr$\mathcal{O}$.

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

Here are the opening paragraphs.

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 a special case of 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.

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

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

Similar to many prior works [AJ15,BV15] building obfuscation from FE, our construction involves a binary tree of FE ciphertexts, yet its structure is slightly different 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.

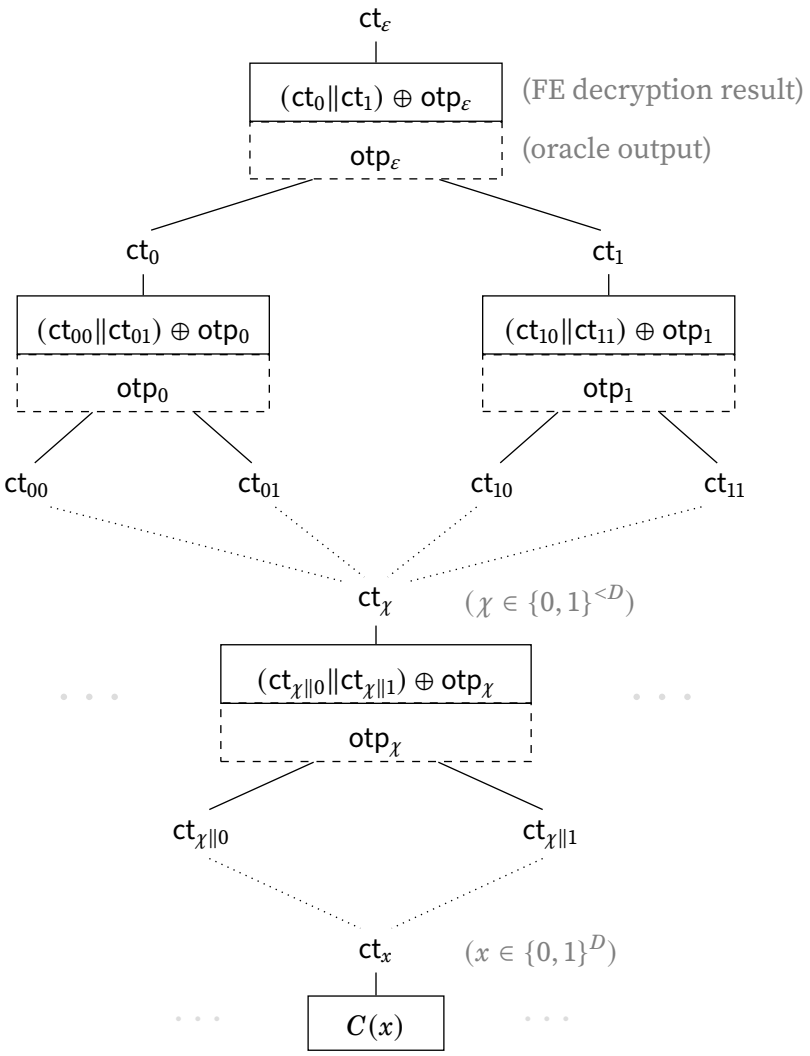

Figure 4. The binary tree of ciphertexts in Construction 1 (normal behavior).

The obfuscation of a circuit C with D-bit input involves a full 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 $\chi \in \{0,1\}^{\leq D}$, 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 $C(x)$.

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:

$$\begin{aligned}(\mathsf{pk}_D, \mathsf{sk}_D) &\xleftarrow{\$} \mathsf{Gen}(1^\lambda, \mathsf{Eval}),\\ (\mathsf{pk}_d, \mathsf{sk}_d) &\xleftarrow{\$} \mathsf{Gen}(1^\lambda, \mathsf{Expand}_d[\mathsf{pk}_{d+1}]) \quad \text{for } d = D-1, \dots, 0,\end{aligned}$$

   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:

$$\begin{aligned}s_\varepsilon &\xleftarrow{\$} \{0,1\}^\lambda, & r_\varepsilon &\xleftarrow{\$} \{0,1\}^\lambda,\\ \mathsf{flag}_\varepsilon &\leftarrow \mathsf{normal}, & \mathsf{info}_\varepsilon &\leftarrow (C, \{k_{i,j}\}_{0\le i<D, 1\le j\le B}, s_\chi),\\ \mathsf{ct}_\varepsilon &\leftarrow \mathsf{Enc}(\mathsf{pk}_0, \mathsf{flag}_\varepsilon, \varepsilon, \mathsf{info}_\varepsilon; r_\varepsilon).\end{aligned}$$

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

$$\begin{aligned}&H(k_{d,1}, \chi_d \| 0^{D-d}) \| \cdots \| H(k_{d,B}, \chi_d \| 0^{D-d})\\ &= \mathcal{O}(\mathsf{hEval}, h_{d,1}, \chi_d \| 0^{D-d}) \| \cdots \| \mathcal{O}(\mathsf{hEval}, h_{d,B}, \chi_d \| 0^{D-d}).\end{aligned}$$

$\mathsf{Expand}_d[\mathsf{pk}_{d+1}](\mathsf{flag}_\chi, \chi, \mathsf{info}_\chi)$ — **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).

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

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.

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

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 $v_\chi \leftarrow$ $G_v(\sigma_{\chi,1})\|\cdots\|G_v(\sigma_{\chi,\beta-1})\|w_\chi$
$\|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_{\beta+1} \oplus H(k_{d,\beta+1}, \chi\|0^{D-d}))\|\cdots$
$\|([\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_B \oplus H(k_{d,B}, \chi\|0^{D-d}))$

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

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.

- For $(\mathsf{hEval}, h_{i,j}, \chi \| 0^{D-i})$, the 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 $\mathcal{S}_1$.

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.

7.2 Hybrids over Levels

To prove Theorem 1, we consider $\mathsf{Hyb}_{\delta,\star}$[9] for $0 \leq \delta \leq D$ 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 $s_{\chi_{\leq\delta}}$;
- $h_{i,j}$ for $i < \delta$ *does not have* a corresponding PRF key;
- $h_{i,j}$ for $i \geq \delta$ *has* a corresponding PRF key;
- ct_χ for $0 < |\chi| \leq \delta$ 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 $|\chi| \geq \delta$ is in *normal* mode and uses *pseudorandom* r_χ expanded from $s_{\chi_{\leq\delta-1}}$.

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* $\mathsf{Hyb}_{\mathrm{real}} \approx_{\mathrm{s}} \mathsf{Hyb}_{0,\$\$}$.

[9] 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.

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	$\{0,1\}^{<D} \times \{1,\dots,B\} \to \{0,1\}^\lambda$
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 $\uparrow$ $\mathcal{S}_3$ $\downarrow$	in simulator
$k \stackrel{?}{=} k_{i,j}$	non-existent
$h_{i,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_j$
$h_{i,j} : t$	$F_{i,j}(t)$

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	$\{0,1\}^{\delta} \to \{0,1\}^\lambda$	$\{0,1\}^{\delta-1} \to \{0,1\}^\lambda$
▸ $\lvert\chi\rvert < \delta$:		
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	$F_r(\chi)$	expand from
s_χ	$F_s(\chi)$	$s_{\chi_{\le\delta-1}} = F_s(\chi_{\le\delta-1})$
▸ $\lvert\chi\rvert \ge \delta$:		
flag_χ	normal	
info_χ	$(C, \{k_{i,j}\}_{\lvert\chi\rvert\le i<D, 1\le j\le B}, s_\chi)$	
(s,r) 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 $\mathsf{Hyb}_{\delta,G(\$)}$
$k \stackrel{?}{=} k_{i,j}$	$\delta \le i < D$, $1 \le j \le B$ (checks for all existent $k_{i,j}$'s)	
▸ $i < \delta$:		
$h_{i,j} : \chi$	$G_v(F_\sigma(\chi,j)) \oplus [\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_j$	
$h_{i,j} : t$	$F_{i,j}(t)$	
▸ $i \ge \delta$:		
$h_{i,j} : \chi$	$H(k_{i,j}, \chi\|0^{D-i})$	
$h_{i,j} : t$	$H(k_{i,j}, t)$	

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* $\mathsf{Hyb}_{D,\$\$} \approx \mathsf{Hyb}_{\mathrm{sim}}$.

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 "≈" 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 $|\chi| = D$:

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

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).

$\{k_{i,j}\}$	$\substack{i=\delta:\ \beta\le j\le B \\ \delta<i<D:\ 1\le j\le B}$		$\substack{i=\delta:\ \beta+1\le j\le B \\ \delta<i<D:\quad 1\le j\le B}$		
$\{F_{i,j}\}$	$\substack{\{0,1\}^D \\ -1mu\to \{0,1\}^L}$ 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	$\substack{(\{0,1\}^{<\delta}\times\{1,\ldots,B\}) \\ -1mu\cup(\{0,1\}^\delta\times\{1,\ldots,\beta-1\})}$ $\to \{0,1\}^\lambda$			$\{0,1\}^\lambda \leftarrow$ $\substack{(\{0,1\}^{<\delta}\times\{1,\ldots,B\}) \\ -1mu\cup(\{0,1\}^\delta\times\{1,\ldots,\beta\})}$	
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	$F_s(\chi)$				
$\boxed{w_\chi}$	$[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta \oplus H(k_{\delta,\beta}, \chi\|0^{D-\delta})$		$\substack{[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta \\ -1mu\oplus F_{\delta,\beta}(\chi\|0^{D-\delta})}$	$F_{\delta,\beta}(\chi\|0^{D-\delta})$	$G_v(F_\sigma(\chi,\beta))$
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 $j = \beta$:					
$h_{\delta,\beta} : \chi$	$H(k_{\delta,\beta}, \chi\|0^{D-\delta})$		$F_{\delta,\beta}(\chi\|0^{D-\delta})$	$\substack{F_{\delta,\beta}(\chi\|0^{D-\delta}) \\ -1mu\oplus[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta}$	$\substack{G_v(F_\sigma(\chi,\beta)) \\ -1mu\oplus[\mathsf{ct}_{\chi\|0}\|\mathsf{ct}_{\chi\|1}]_\beta}$
$h_{\delta,\beta} : t$	$H(k_{\delta,\beta}, t)$		$F_{\delta,\beta}(t)$	$F_{\delta,\beta}(t)$	$F_{\delta,\beta}(t)$
▸ $i = \delta$ 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 $0<\varepsilon<1/2$.[10] We have (below, $|\cdot|$ is the bit length of everything)

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

[10] 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

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

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.

[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, *3rd Conference on Information-Theoretic Cryptography (ITC 2022)*, volume 230 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:21, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[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-Körner 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.

[JLL22] Aayush Jain, Huijia Lin, and Ji Luo. Personal communication, 2022.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 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.

[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 John Pippenger and Michael John Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, apr 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.