On the Impossibility of Approximate Obfuscation and Applications to Resettable Cryptography

Nir Bitansky* and Omer Paneth†

Tel Aviv University and Boston University

December 31, 2012

Abstract

The traditional notion of *program obfuscation* requires that an obfuscation $\tilde{f}$ of a program f computes the exact same function as f, but beyond that, the code of $\tilde{f}$ should not leak any information about f. This strong notion of *virtual black-box* security was shown by Barak et al. (CRYPTO 2001) to be impossible to achieve, for certain *unobfuscatable function families*. The same work raised the question of *approximate obfuscation*, where the obfuscated $\tilde{f}$ is only required to approximate $\tilde{f}$; that is, $\tilde{f}$ only agrees with f on some input distribution.

We show that, assuming *trapdoor permutations*, there exist families of *robust unobfuscatable functions* for which even approximate obfuscation is impossible. That is, obfuscation is impossible even if the obfuscated $\tilde{f}$ only agrees with f with probability slightly more than $\frac{1}{2}$, on a uniformly sampled input (below $\frac{1}{2}$-agreement, the function obfuscated by $\tilde{f}$ is not uniquely defined). Additionally, we show that, assuming only one-way functions, we can rule out approximate obfuscation where $\tilde{f}$ is not allowed to err, but may refuse to compute f with probability close to 1.

We then demonstrate the power of robust unobfuscatable functions by exhibiting new implications to resettable protocols that so far have been out of our reach. Concretely, we obtain a new non-black-box simulation technique that reduces the assumptions required for resettably-sound zero-knowledge protocols to *one-way functions*, as well as reduce round-complexity. We also present a new simplified construction of simultaneously resettable zero-knowledge protocols that does not rely on collision-resistent hashing. Finally, we construct a three-message simultaneously resettable WI *argument of knowledge* (with a non-black-box knowledge extractor). Our constructions are based on a special kind of "resettable slots" that are useful for a non-black-box simulator, but not for a resetting prover.

*This research was done while visiting Boston University and IBM T. J. Watson Research Center. Supported by the Check Point Institute for Information Security, an ISF grant 20006317, and the Fulbright program.

†Supported by an NSF grant 1218461.

Contents

1 Introduction

The problem of *program obfuscation* concerns the task of rewriting programs in a way that makes their code "unintelligible" without destroying its functionality. The rigorous study of the problem was initiated in the work of Barak et al. [BGI+01], which formalize secure obfuscation according to the *virtual black-box* notion. At high-level, this notion implies that whatever an *efficient* learner can deduce, given an obfuscation $\tilde{f}$ of a program f, should also be learnable, given only black-box access to f. The same work shows, however, that in general, this notion may not be achievable. More concretely, [BGI+01] show that there exist an *unobfuscatable* family of functions $\{f_{\mathsf{k}}\}$, for which *any* program $\tilde{f}_{\mathsf{k}}$ that computes the same function as f_{k} leaks information that cannot be learned, given only black-box access to f_{k}, assuming k is chosen at random.

Approximate obfuscation. In light of the [BGI+01] impossibility, several followup works studied notions of obfuscation with relaxed security [AW07, GR07, HMLS07, HRSV07, BC10]. A different kind of relaxation is to weaken the functionality (rather than security) requirement. In this context, Barak et al. [BGI+01] put forth the notion of *approximate obfuscation*, where the obfuscated program $\tilde{f}$ is not necessarily required to perfectly compute the same function as f, but only to approximate f in some sense. Concretely, Barak et al. show that obfuscation of general programs is impossible if an obfuscation $\tilde{f}$ is required to approximate f in the following strong sense: for every input, the obfuscation $\tilde{f}$ is allowed to err only with negligible probability, over the coins of the obfuscator algorithm that samples $\tilde{f}$.

However, this notion of approximation is rather strong, since it provides a correctness guarantee **for every** input; in particular, if we allow the obfuscation $\tilde{f}$ to err even on $\mathrm{poly}(n)$-many inputs, no impossibility is known. In fact, one may require that $\tilde{f}$ only agrees with the function f, with high-enough probability, over inputs drawn from some specific distribution D, such as the uniform distribution. Indeed, this relaxation may suffice for many applications: for example, protecting the secret key of a program that digitally signs messages from a high-entropy distribution, or protecting the code of an algorithm that can factor random integers. The existence of such approximate obfuscators was left by Barak et al. as an open question.

1.1 The Impossibility Result

We show that approximate obfuscation of general programs is impossible. Concretely, we construct *error-robust* unobfuscatable functions that rule out the approximate obfuscation notion discussed above. (where $\tilde{f}$ is only required to often agree with f on a uniformly random input, and may otherwise err). To construct error-robust unobfuscatable functions, we study a weaker notion of *robust* unobfuscatable functions. Such function only rule out a stronger notion of obfuscation where $\tilde{f}$ is limited to making only *detectable errors* (e.g., output $\perp$). Although weaker than their error-robust counterparts, robust unobfuscatable functions will play a significant role as a building block for error-robust functions, and will also allow us to reduce computational assumptions in some of our results.

Theorem 1.1 (existence of robust unobfuscatable functions - informal)**.**

1. *Assuming the existence of one-way functions, there exist robust unobfuscatable families.*
2. *Assuming the existence of trapdoor permutations, there exist error-robust unobfuscatable families.*

We further discuss the two notions of robustness and the corresponding constructions in Section 1.3.

1.2 Applications to Non-Black-Box Simulation and Resettable Cryptography

We demonstrate the power of robust unobfuscatable functions by exhibiting new implications to resettable protocols. Indeed, the recent work of [BP12] shows that the impossibility of obfuscation also has

positive implications to ZK protocols with non-black-box simulation. Specifically, they show a construction of *resettably-sound* ZK using unobfuscatable functions. Based on ideas from [BP12], and on our construction of error-robust unobfuscatable functions, we present a new non-black-box simulation technique, and use it to get simplified constructions of *resettably sound* ZK and *simultaneously resettable* ZK, with weaker computational assumptions, and reduced round-complexity.

We now further discuss the model of resetting, explain the related challenges, and present the concrete improvements we obtain.

Background. Resettable cryptography deals with the problem of secure computation in the presence of malicious parties that can *reset* honest paries, forcing them to repeat any execution from the same initial state and random tape, whereas the malicious party may use different inputs and messages. This setting is motivated by scenarios in which cryptographic protocols are run by parties that cannot regenerate fresh randomness or keep a state between different executions, occurring after different reset attempts. Common examples include: parties implemented on stateless hardware, inside virtual machines, or parties that are required to perform multiple consistent executions in a distributed environment.

Resettable protocols were first studied by [CGGM00] in the context of zero-knowledge (ZK). They define and construct *resettable ZK* proof systems, requiring that the view of any malicious verifier in a proof can be efficiently simulated, even if the verifier is allowed to reset the honest prover. [BGGL01] define and construct *resettably-sound* ZK protocols guaranteeing soundness, even against malicious provers that can reset the honest verifier. [DGS09] construct *simultaneously resettable ZK* protocols, i.e., protocols that are both resettable ZK and resettably-sound. Resettable soundness and simultaneous resettability subsequently had a major role in the construction of resettable secure computation protocols for more general functionalities [GS09, GM11]. (In addition, several works tr to construct improved rZK and rsZK protocols in the relaxed *bare public-key model* [CGGM00, MR01, DFG$^+$11]; in this work, however, we focus on the plain model.)

The challenge of resetting, in a nutshell. In standard (non-resetting) ZK protocols, simulation is, traditionally, performed by "rewinding" (or in other words, resetting) the verifier. By rewinding the verifier, the simulator can extract from the verifier the required information for generating a simulated proof, without knowing a corresponding witness. As shown by [CGGM00], such rewinding techniques (concretely [RK99]) can also be extended to simulate resetting verifiers, yielding resettable ZK protocols; on the other side, rewinding techniques alone cannot be enough for simulation in resettably-sound protocols. Indeed, any rewinding strategy applied by the simulator can also be applied by a malicious resetting prover, in which case soundness cannot be guaranteed (for non-trivial languages). Thus, resettable soundness is impossible to achieve with a simulator that only uses the verifier as a black-box, and indeed to construct such protocols [BGGL01] rely on Barak's non-black-box simulation technique [Bar01].

Comparing to black-box ZK protocols (e.g., [FS89]), Barak's ZK protocol requires stronger assumptions (collision-resistant hashing vs. one-way functions), more rounds (eight-message vs. four-message), and rather heavy machinery (PCPs). The resettably-sound protocol of [BGGL01], which relies on Barak's technique, inherits all of the above. Naturally, the situation is even harder when trying to apply Barak's technique to achieve simultaneous resettability; indeed, to obtain a simultaneous resettable protocol, [DGS09] provide a highly non-trivial extension of Barak's technique that is carefully combined with the rewinding techniques required to obtain resettable ZK.

Resettable protocols via robust unobfuscatable functions. Extending [BP12], we show a new non-black-box simulation technique yielding new and improved constructions of resettably-sound and simultaneously resettable protocols. The constructions are based on a unified paradigm that couples non-black-box simulation with existing black-box simulation techniques. Moreover, our resettably-sound protocol relies on weaker assumptions and has fewer rounds than previous protocols. More concretely, we obtain the following:

- A resettably-sound ZK protocol based on one-way functions. (This was also achieved in a recent work by Chung, Pass, and Seth [CPS12]. Previous constructions relied on collision-resistent hashing or oblivious transfer[BGGL01, BP12]). As an immediate corollary from the work of [BGGL01], we also get a resettable ZK argument of knowledge based on one-way functions.
- A resettably-sound concurrent ZK protocol based on one-way functions. The protocol can be transformed to a simultaneously-resettable protocol, with the additional assumption of trapdoor permutations. (Previous constructions relied on trapdoor permutations and collision-resistant hashing [DGS09].)
- A 6-message resettably-sound ZK protocol, based on trapdoor permutations, and a 4-message protocol based on fully homomorphic encryption. (Previous constructions required eight messages [BGGL01].[1])
- A 3-message simultaneously-resettable WI argument of knowledge, based on trapdoor permutations. (Previous constructions relied also on collision-resistant hashing, and required at least ten messages [COSV12].)

An additional interesting feature is that our protocols do not make any use of heavy machinery, such as PCPs and may thus be more efficient in practice. (PCPs were also circumvented in [BP12], but only for resettable-soundness.)

1.3 Constructing Robust Unobfuscatable Functions

Our first (and arguably, harder) step is to construct robust unobfuscatable functions from one-way functions. Our second step is to compile any robust unobfuscatable family into an error-robust family using trapdoor permutations. We now overview the ideas behind the constructions; we start by a more precise presentation of robust and error-robust unobfuscatable functions.

Informally, a function family $\mathcal{F}$ is a(n) (error-)robust unobfuscatable family if:

1. Efficient learners, with black-box access to a random $f_{\mathsf{k}} \leftarrow \mathcal{F}$, cannot learn k.
2. There exists a samplable input distribution D, and an efficient extractor that extracts k from any circuit C that approximates f_{k} in the following sense:
   - If $\mathcal{F}$ is **robust**, C is required to agree with f_{k}, on inputs drawn from D, with probability $\frac{1}{\mathrm{poly}(n)}$, but is only allowed to make detectable errors, i.e., it either outputs the correct answer or $\perp$.
   - If $\mathcal{F}$ is **error-robust**, C is required to agree with f_{k}, on inputs drawn from D, with probability $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$, but is allowed to err on the rest of the space.

Crucially, we require that the distribution D is publicly samplable and independent of the key k; specifically, this implies that extraction succeeds even if a circuit fails to compute f_{k} on points that depend on the key k. Also, we shall consider a variant of the above definition where some unlearnable property $P(\mathsf{k})$ is extracted, rather than the entire key k; this will be sufficient for all of our applications. Finally, we note that in the above definition we choose to consider extractors that output a unique key k from a given circuit C, and thus we require that C approximates a single function from the family. This is why we require, in the case of error-robustness, more than $\frac{1}{2}$-agreement (otherwise, for any function family with more than a single function, there exist circuits that can approximate two different functions, with probability $\frac{1}{2}$ each). (We also discuss more liberal definitions.)

The starting point for our construction of robust unobfuscatable functions is the construction of (non-robust) unobfuscatable functions from [BGI$^+$01], and can be seen as a "random-self-reducible" version of it. We start by describing the construction of [BGI$^+$01], and then move on to describe the main modifications that we introduce.

[1] Recently, Ostrovsky and Visconti [OV12] showed that the [BGGL01] protocol can be compressed into 6-messages.

The construction of [BGI$^+$01]. Barak et al. [BGI$^+$01] construct a family $\mathcal{F} = \{f_{\mathsf{k}}\}$ of unobfuscatable functions as follows. The key k consists of two random strings (a, b) and a key sk for a symmetric encryption scheme. For a simpler exposition, let us assume for now that the encryption scheme is *fully homomorphic*. Later, we will explain how to modify the construction to rely on CCA-1 symmetric encryption (that can be based on one-way functions).

We also describe $f_{\mathsf{k}} = f_{a,b,\mathsf{sk}}$ as a randomized function; however, it could be made deterministic using a pseudo-random function. The function is defined as follows:

1. On input a, output b.
2. On input "Encrypt", output an encryption of a.
3. On input that is an encryption of b, output b.

Notice that given black-box access to the function, the only way to learn b is to first learn a, which will break the security of the encryption scheme, or produce an encryption of b, which is information theoretically impossible, conditioned on not learning a. On the other hand, given a circuit C that computes $f_{a,b,\mathsf{sk}}$, we can obtain an encryption of a, evaluate the circuit C homomorphically on this encryption, and obtain an encryption of b, and then learn b using another black-box application of C.

Making the construction "random-self-reducible". The latter construction is not robust: for example, given a circuit C that computes $f_{a,b,\mathsf{sk}}$ but only errs on the single input a, we can no longer extract b. At high-level, to to overcome this problem, we use a pseudo-random function PRF to "encode" the connection between a and b in random inputs. Specifically, we add the seed s of the PRF to the key of f and modify it to also evaluate the two functions $\mathbb{G}(x) = \mathsf{PRF}_{\mathsf{s}}(x)$ and, $\mathbb{G}'(x) = b \oplus \mathsf{PRF}_{\mathsf{s}}(x \oplus a)$. Now, for a random string r, we can query $\mathbb{G}$ with r and $\mathbb{G}'$ with $r \oplus a$ and obtain $\mathbb{G}(r) \oplus \mathbb{G}'(r \oplus a) = b$. Still, without knowing a, an efficient algorithm that gets black-box access to $\mathbb{G}$ and $\mathbb{G}'$ cannot find any correlation between the functions. The gain is that now each individual query is distributed uniformly and therefore we can learn b from a, even if the function errs on, say, a $\frac{1}{4}$-fraction of inputs. To extract from circuits that make more errors, we change f to evaluate the $\mathbb{G}$ on many random inputs in parallel, amplifying the probability of getting the correct answer on a single random input.

In order to successfully extract b, it is not sufficient to deal with circuits that err on a, but we should also handle circuits that err given encryptions of b, i.e. they do not output b. We would like to use a similar idea to the one described before; namely, include a function $\mathbb{G}''(\mathsf{c})$ that, on input c, decrypts c to some plaintext x, and outputs $b \oplus \mathbb{G}_{\mathsf{s}}(x \oplus b)$. However, given black-box access to such a function, semantic security is not maintained. Indeed, two encryptions c_1 and c_2 correspond to the same plaintext *iff* $\mathbb{G}''(\mathsf{c}_1) = \mathbb{G}''(\mathsf{c}_2)$, and thus it is possible to learn a and thus also b in a black-box way. We show that this can be solved by using a *symmetric* encryption scheme with public randomness (where the encryption algorithm outputs its coins in the clear). Specifically, denoting the public randomness in c by $\mathsf{pub}(\mathsf{c})$, we redefine $\mathbb{G}''(\mathsf{c}) = b \oplus \mathsf{PRF}_{\mathsf{s}}(x \oplus b || \mathsf{pub}(\mathsf{c}))$, where x is the decryption of c. Note that if $\mathsf{c}_1 \neq \mathsf{c}_2$, and $x_1 = x_2$, then $\mathsf{pub}(\mathsf{c}_1) \neq \mathsf{pub}(\mathsf{c}_2)$; therefore, also $\mathbb{G}''(\mathsf{c}_1)$ and $\mathbb{G}''(\mathsf{c}_2)$ are distributed pseudo independently at random, and we can show that semantic security is maintained. However, given a random encryption c_1 of b, it is possible to learn b by first sampling a random string r and a random encryption c_2 of r, then homomorphically computing an encryption c_3 of $b \oplus r$, and finally, evaluating $\mathbb{G}''(\mathsf{c}_3) \oplus \mathsf{PRF}_{\mathsf{s}}(r || \mathsf{pub}(\mathsf{c}_3)) = b$. The input $r || \mathsf{pub}(\mathsf{c}_3)$ to $\mathsf{PRF}_{\mathsf{s}}$ is distributed uniformly independently b. (We note that the public randomness requirement can be relaxed to a slightly more complicated "decomposability requirement" that can be obtained from any encryption scheme).

Getting rid of homomorphic encryption. The construction of [BGI$^+$01] only uses a standard symmetric encryption, and homomorphic operations are performed gate by gate by f_{k}: given two encrypted input bits, it decrypts them, evaluates the desired operation and returns the result encrypted. Since such queries consist of random encryptions, we may hope that a circuit C that does not err too much will correctly compute extractor's homomorphic operation with high probability. However, this is not the case since in a long (a-priori unbounded) homomorphic computation, errors accumulate, and there is no

way of checking consistency along the way.

Here, we use the fact that, in robust unobfuscatable functions, we can assume that C only makes "detectable errors"; we show that this is enough in order to recover from errors during the homomorphic computation. Specifically, when a single homomorphic operation fails (the circuit outputs $\perp$), we would like the extractor to rerandomize the input encryptions and try again. This could be achieved by relying on rerandomizable encryption. However, we wish to avoid this assumption and provide a solution based on one-way functions. For this purpose, we modify f_{k} to return many random encryptions of the resulting bit, and show a procedure that assures that the extractor is always left with enough "good" encryptions to continue the homomorphic evaluation. (Formally, we will need to use "invoker randomizable" PRFs [BGI$^+$01]; however, this is not needed for understanding the high-level idea behind the construction.)

From several (key-dependent) distributions to one samplable independent distribution. The family $\mathcal{F}$ described above, corresponds to $d = O(1)$ input distributions $D_1, \ldots, D_d$, where if a circuit C approximates f_{k} on every distribution D_i with sufficient probability, extraction succeeds. Specifically, the input distributions correspond to the different functions implemented by f_{k}: $\mathbb{G}, \mathbb{G}', \mathbb{G}''$ and the homomorphic evaluation functionality. In particular, for the homomorphic evaluation functionality, we need a distribution for each configuration of possible gate and input plaintexts. The problem is that the latter distributions may not be directly samplable without the secret encryption key. Indeed, in symmetric-key encryption (based on one-way functions), we do not know how to sample encryptions of individual bits, without the secret key.

Instead, we use an encryption scheme that supports sampling ciphers of random plaintexts, independently of the secret key. At high-level, if the circuit does not abort given an encryption of a random bit, with sufficiently high-probability, then it also does not abort for encryptions of specific bits, with related probability. To make sure that the probability is indeed high-enough we use again parallel repetition. (Note that, when emulating a homomorphic operation, the extractor queries f_{k} on encryptions of individual bits; however, it does not sample those encryptions on its own, but rather these were obtained as the result of a previous homomorphic operation.) Eventually, we can bundle all of the distributions in to one distribution that is samplable independently of the secret key for f_{k}; furthermore, for our construction, this distribution can simply be the *uniform* one.

Necessity of one-way functions. We also show that one-way functions are not only sufficient, but also necessary for robust unobfuscatable functions. (The one-way function simply maps $(\mathsf{k}, x_1, \ldots, x_m)$ to $(f_{\mathsf{k}}(x_1), \ldots, f_{\mathsf{k}}(x_m), x_1, \ldots, x_m)$, for $m \gg |\mathsf{k}|$.) This should be contrasted with non-robust unobfuscatable functions, where a similar implication is not known; in fact, [BGI$^+$01] construct (non-robust) unobfuscatable Turing machine (rather than circuit) families without any computational assumptions, whereas robust unobfusactable functions also imply one-way functions in the Turing machine case.

Error-robust unobfuscatable functions. We show how to transform any robust unobfuscatable function family into an error-robust unobfuscatable family. Recall that, for an unobfuscatable function family to be error-robust, we require that for any f_{k}, it is possible to extract k (or some unlearnable property of k), from any circuit C that agrees with f_{k} with probability noticeably greater than $1/2$. On inputs where C and f_{k} disagree there are no restrictions on the output of C (unlike the robust case, where the output had to be $\perp$).

The idea is to first construct a stronger type of robust unobfuscatable functions called *verifiable robust* unobfuscatable functions. Such functions also have a public verification key vk that uniquely defines f_{k}; however, the function is still black-box unlearnable, even given vk. Moreover, vk allows verifying that an answer, given by a circuit C, is consistent with f_{k}, We construct such functions based on non-interactive commitments and 2-message witness indistinguishable proofs (ZAPs). Specifically, the verification key, consists of a commitment to a robust unobfuscatable function f_{k}, and the outputs of the verifiable function include a ZAP showing consistency with the committed f_{k}.The actual construction

is a bit more involved: to guarantee that the ZAP does not reveal the secret key k, we evaluate two functions in parallel with independent keys and prove that one of them was computed correctly. The construction uses ideas from [FS90, COSV12].

Equipped with verifiable robust unobfuscatable functions, we can construct error-robust unobfuscatable functions by appending to each output the public key. This allows an extractor to identify the verification key vk returned with probability greater than half, and thus obtain a circuit that either computes f_{k} or outputs $\perp$. Answers containing a verification key different from vk, or answers that does not verify under vk are treated as $\perp$.

Verifiable robust unobfuscatable functions will also set as a convenient abstraction in the construction of resettable protocols.

Verifiable robust unobfuscatable functions and digital signatures. Interestingly, verifiable robust unobfuscatable functions are equivalent to a unique kind of signature schemes that have a weak unforgebility guaranty, but a strong extraction guaranty. Specifically, it is guaranteed that no adversary can forge a signature on a random message $m \leftarrow U$, even if it is given access to a signing oracle prior to receiving m; however, given the circuit of an adversary that manages to sign random messages with noticeably probability, it is possible to efficiently extract the secret signing key. The transformation between the two is rather direct: the private signing key and public verification key are simply the secret key k and public verification key vk of a verifiable robust unobfuscatable function; signing a message m is simply done by computing $f_{\mathsf{k}}(m)$, and the verification is identical to that of the verifiable unobfuscatable function. Indeed, it is not hard to see that the non-black-box learnability and black-box unlearnability of the verifiable robust family $\{f_{\mathsf{k}}\}$, are equivalent to the above extraction and unforgebility properties.

1.4 Applications to Resettable Protocols

In the world of black-box ZK, simulation is performed by "rewinding" the verifier. A common paradigm for designing ZK with black-box simulation is using the notion of a "slot". A slot is a two message subprotocol, where the prover sends a query and the verifier sends a response. Slot-based protocols are designed such that: (a) a rewinding simulator that can get answers to different queries in the slot, can use them to obtain a "trapdoor" that allows generating simulated proofs (without the witness), and (b) a cheating prover that participates in a slot interaction only once cannot obtain the trapdoor and cannot break the soundness of the protocol. Standard constructions of stand-alone ZK protocols are based on a constant number of slots, whereas known concurrent or resettable zero-knowledge protocols require a super-constant number of slots.

When constructing resettably-sound ZK, the idea of slots seems to lose its appeal since a resetting prover can rewind a slot, learn the trapdoor, and thus break soundness. However, we may hope to design a special *resettable slot* that is useful for resettably-sound protocols. A resetting cheating prover that can rewind the slot many times should not obtain the trapdoor, while, a simulator can use the code of the cheating verifier implementing the slot to obtain a trapdoor, and simulate a proof. This motivation was already raised in the work of [DGS09] while trying to construct a resettable slot to get simultaneously resettable ZK. Eventually, the techniques presented by [DGS09] achieved the goal of simultaneously resettable ZK, but diverged from the idea of a resettable slot.

Resettable slots from verifiable robust functions. Verifiable robust unobfuscatable functions provide a natural and simple construction of a resettable slot. Given a public verification key vk for such a function f_{k}, the slot query q is just a query to f_{k}, and the slot answer is just $f_{\mathsf{k}}(q)$. The trapdoor is defined to be the key k of f_{k} (or some unlearnable property of k). Since k is hard to learn, given black box access to a random f_{k}, it follows that a resetting prover that rewinds the slot cannot obtain a trapdoor. However, given the code of any verifier that provides valid answers in the slot with significant probability (where validity can be checked using vk), the unlearnable k (and thus a trapdoor) can be obtained.

Comparison to [BP12]. In the work of [BP12], unobfuscatable functions that are not robust are used to get a resettably-sound ZK protocol. The problem with using non-robust functions is that a malicious verifier evaluating the unobfuscatable function may introduce errors or simply refuse to compute the function on specific inputs. [BP12] solved this problem by letting the prover and verifier execute a *secure function evaluation* protocol, in order to compute the function. However, constructing such protocols that are secure in the resetting model turns out to be challenging and introduces more assumptions, rounds, and complexity to the protocol. Relying on robust unobfuscatable functions, this problem completely disappears.

Constructing simultaneously resettable ZK. By plugging our resettable slot into the concurrent ZK protocol of Richardson and Kilian [RK99], we get a concurrent ZK protocol that is also resettably-sound. This is, in fact, "the hard part" of constructing simultaneously resettable protocols; once this is obtained, the general transformation of [DGS09] can be applied to obtain full-fledged simultaneous resettability. Our protocols rely only on trapdoor permutations, in contrast to previous constructions that relied additionally on collision-resistent hashing. (In fact, our resettably-sound concurrent ZK protocol can be constructed from one-way functions, and thus any improvement in assumptions for the [DGS09] transformation, will also result in an improvement for our protocol.)

An open question in the context of simultaneously resettable ZK, which we find fascinating , is to construct protocols with a logarithmic round-complexity (matching the what is known for resettable ZK [PRS02]). Unfortunately, we cannot plug our resettable slot into the concurrent ZK protocol of [PRS02] since this protocol requires that the ratio between the time it takes to evaluate the slot and the time it takes to extract from the slot is constant. In our construction, the extraction time may be quadratic in the evaluation time. This ratio can be improved to depend only on the security parameter assuming homomorphic encryption; however, we still do not know how to construct unobfuscatable function, for which this ratio is constant.

Constructing round-efficient resettably-sound ZK. By plugging our resettable slot into the round-efficient ZK protocol of Feige and Shamir [FS89], we may hope to get a 4-message resettably-sound ZK protocol. However, as already observed in [BP12], using just one slot requires that the ratio between the time it takes to evaluate the slot, and the time it takes to extract from the slot, is a fixed polynomial in the security parameter. To fix the problem, we can add a second slot to our protocol resulting in a 6-message protocol. Alternatively, assuming homomorphic encryption, we can construct a resettable slot where extraction is only slower than evaluation up to a fixed $\mathrm{poly}(n)$ factor. Such a construction will result in a 4-message resettably-sound ZK protocol.

Constructing resettably-sound ZK from minimal assumptions. Our constructions of error-robust unobfuscatable functions relies on non-interactive commitments and 2-message witness indistinguishable proofs (ZAPs [DN07]); however, in order to construct a resettably-sound protocol, it in fact suffices to use robust (rather than error-robust) unobfuscatable functions, which can already be constructed from one-way functions. Concretely, the corresponding protocol follows the transformation from robust to error-robust unobfuscatable functions, but uses interactive (2-message) commitments [Nao91], instead of non-interactive ones, and instance-dependent resettable witness-indistinguishable arguments based on one-way functions, instead of ZAPs.

Recently, Chung, Pass, and Seth [CPS12] also constructed resettably-sound ZK based on one-way functions. Specifically, they show an elegant way of replacing collision-resistant hashing in Barak's non-black-box protocol [Bar01] with strong digital signatures, and then apply the [BGGL01] transformation.

3-message simultaneously-resettable witness indistinguishability. We also construct a 3-message simultaneously-resettable WI argument of knowledge protocol. (Indeed, notice that while ZAPs are 2-message simultaneously-resettable WI, they are not known to also be an argument of knowledge, without relying on non-standard "knowledge assumptions".)

Previously, Cho et al. [COSV12] constructed a simultaneously-resettable WI argument of knowl-

edge based on trapdoor permutations and collision-resistant hashing. The round complexity of their protocol is bounded from below by that of the [BGGL01] protocol, which is used as a subroutine; over all the protocol requires more than ten messages. Our new protocol is based on robust unobfuscatable functions and only assumes trapdoor permutations. The protocol has a non-black-box knowledge extractor.

2 Robust Unobfuscatable Functions

In this section, we define the basic notion of robust unobfuscatable functions.

Definition 2.1 ((ϵ, D)-approximation)**.** *Let D be a distribution on $\{0,1\}^n$, a circuit C is said to (ϵ, D)-approximate a function $f : \{0,1\}^n \to \{0,1\}^*$ if:*

1. **It partially agrees with f on inputs drawn from D:**

$$\Pr_{q \leftarrow D}[C(q) = f(q)] \geq \epsilon \ ,$$

2. **It only makes detectable errors on inputs from D:**

$$\Pr_{q \leftarrow D}[C(q) \notin \{\perp, f(q)\}] \leq \mathrm{negl}(n) \ .$$

We say that C (ϵ, D)-approximates f **with errors** *if the second item is not guaranteed.*

Remark 2.1 (abusing notation). Formally, the definition is also parameterized by the negligible function negl bounding the probability of undetectable errors (i.e., $C(q) \notin \{f_{\mathsf{k}}(q), \perp\}$); however, to lighten notation, we use (ϵ, D) rather than $(\epsilon, \mathrm{negl}, D)$. Accordingly, the following definitions are also implicitly parameterized by the same negligible function negl. Eventually, in our applications, we will treat specific families of circuits $C = \{C_n\}$ each with an associated error function negl.

In addition, ϵ and D may depend on n. To lighten notation, we shall often avoid explicitly writing n in their description. In particular, whenever n is clear from the context, we may denote a distribution $\mathcal{D}(1^n)$ by its sampler $\mathcal{D}$.

Definition 2.2 (Robust unobfuscatable functions)**.** *A function family $\mathcal{F} = \{f_{\mathsf{k}}\}_{\mathsf{k} \in \{0,1\}^n, n \in \mathbb{N}}$ is a robust unobfuscatable family, with respect to efficient relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$, if it is:*

1. **Black-box unlearnable:** *For any poly-size learner $\mathsf{L} = \{\mathsf{L}_n\}_{n \in \mathbb{N}}$, and all large enough $n \in \mathbb{N}$:*

$$\Pr_{\mathsf{k} \leftarrow \{0,1\}^n}[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{L}_n^{f_{\mathsf{k}}}] \leq \mathrm{negl}(n) \ .$$

$\mathcal{R}_{\mathcal{F}}$ is thus called "the unlearnable relation".

2. **Non-Black-box learnable:** *There exists an efficient extractor E such that for any noticeable function $\epsilon(n) = n^{-O(1)}$, any large enough $n \in \mathbb{N}$, any $\mathsf{k} \in \{0,1\}^n$, and every circuit C that $(\epsilon, \mathcal{D})$-approximates f_{k}, E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from C:*

$$\Pr_{\mathsf{E}}\left[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n, 1^{1/\epsilon})\right] \geq 1 - \mathrm{negl}(n) \cdot \mathrm{poly}(|C|) \ ,$$

where $\mathcal{D} = \mathcal{D}(1^n)$, $\epsilon = \epsilon(n)$, and poly is a fixed polynomial that depends only on E.

We say that the family is (just) ϵ-robust, for some function $\epsilon = \epsilon(n)$, if there exists an extractor $\mathsf{E} = \mathsf{E}_\epsilon$, such that for all large enough $n \in \mathbb{N}$, $\mathsf{k} \in \{0,1\}^n$, and circuit C that $(\epsilon, \mathcal{D})$-approximates f_{k}:

$$\Pr_{\mathsf{E}}[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n)] \geq 1 - \mathrm{negl}(n) \cdot \mathrm{poly}(|C|) \ .$$

We next define error-robust unobfuscatable functions which are defined analogously, only that extraction should work for any circuit C that sufficiently agrees with f_{k}, even if C makes undetectable errors.

Definition 2.3 (Error-robust unobfuscatable functions)**.** *A family of functions $\mathcal{F} = \{f_{\mathsf{k}}\}_{\mathsf{k}\in\{0,1\}^n, n\in\mathbb{N}}$ is an error-robust unobfuscatable family with respect to efficient relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$ if it is:*

1. **Black-box unlearnable:** *As in Definition 2.2.*
2. **Non-Black-box learnable:** *There exists an efficient extractor E such that for any noticeable function $\epsilon = \epsilon(n)$, all large enough $n \in \mathbb{N}$, any $\mathsf{k} \in \{0,1\}^n$, E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from any circuit C that $(\frac{1}{2}+\epsilon, \mathcal{D})$-approximates f_{k}* **with errors** *(see Definition 2.1):*

$$\Pr_{\mathsf{E}}\left[(\mathsf{k},\mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n, 1^{1/\epsilon})\right] \geq 1 - \mathrm{negl}(n)\cdot \mathrm{poly}(|C|) \ ,$$

   where $\mathcal{D} = \mathcal{D}(1^n)$, $\epsilon = \epsilon(n)$, and poly *is a fixed polynomial that depends only on* E.

Remark 2.2 (The sampler $\mathcal{D}$). In the above definitions, we allow the input sampler $\mathcal{D}$ to represent an arbitrary samplable distribution, where sampling is independent of the key k for the unobfuscatable function. We can consider a more strict (but natural) definition, where $\mathcal{D}(1^n)$ represents the uniform distribution over stings in $\{0,1\}^{\mathrm{poly}(n)}$. Indeed, our constructions also achieve this variant (see Remark 3.1).

Remark 2.3 (Uniquely determined functions). For the case of erroneous circuits, i.e. with non-detectable errors (Definition 2.3), we naturally require that the circuit C determines one specific function, and hence we require more than $\frac{1}{2}$-agreement (which already guarantees that C cannot simultaneously approximate two functions in the family, assuming that any two functions in the family have at most $\frac{1}{2}$-agreement.) In principle, one may also consider alternative definitions of error-robustness where the circuit may somewhat agree with several functions and the extractor is required to extract the keys of all the functions whose agreement with C crosses a given threshold. We restrict attention to the unique-function case, although our techniques naturally extends to such alternative notions.

Remark 2.4 (Unique-witness unlearnable relations). A natural requirement regarding the unlearnable relation $\mathcal{R}_{\mathcal{F}}$ of a family $\mathcal{F}$ is that, given k, there is a unique $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$, and this z can be efficiently computed; this property is satisfied by our constructions, and is required for some of our applications.

2.1 Robustness from Weaker Robustness

In this section, we discuss two natural relaxations of robust obfuscation that will be convenient to work with, and show how to transform functions that are unobfuscatable according to these notions to functions that are robust according to Definition 2.2. We restrict attention to approximation **without errors**. (Eventually, in Section 4 we show how to go from robust functions to error-robust ones.)

We first show that, using parallel repetition, we can always amplify the robustness of an unobfuscatable family (at the cost of blowing up the input and output size by a factor of n).

Lemma 2.1. *Any $\left(1-\frac{1}{\sqrt[4]{n}}\right)$-robust $\mathcal{G}$ can be transformed to a robust $\mathcal{F}$, where the size of inputs and outputs grows by a factor of n.*

We describe the construction behind the lemma, followed by the analysis.

Construction 2.1 (Robustness from $\left(1-\frac{1}{\sqrt[4]{n}}\right)$-robustness). Let $\mathcal{G}$ be a $\left(1-\frac{1}{\sqrt[4]{n}}\right)$-robust unobfuscatable family with respect to an unlearnable relation $\mathcal{R}_{\mathcal{F}}$, and an input distribution ensemble $\mathcal{D}$; We construct a new robust unobfuscatable family $\mathcal{F}$ with respect to the same relation $\mathcal{R}_{\mathcal{F}}$, and the n-fold distribution ensemble $\widehat{\mathcal{D}} = \mathcal{D}\times\cdots\times\mathcal{D}$. Any function $f_{\mathsf{k}} : \{0,1\}^{n^2} \to \{0,1\}^{n^2}$ is just the n-fold version $\widehat{g_{\mathsf{k}}}(q_1,\ldots,q_n) = (g_{\mathsf{k}}(q_1),\ldots,g_{\mathsf{k}}(q_n))$ of the function $g_{\mathsf{k}} : \{0,1\}^n \to \{0,1\}^n$.

Proof sketch of Lemma 2.1. Unlearnability of $\mathcal{F}$ follows directly from that of $\mathcal{G}$, since any oracle calls $f_{\mathsf{k}} \in \mathcal{F}$ can be simulated by n oracle calls to the corresponding g_{k}. To show that the function is robust, we describe its extractor $\widehat{\mathsf{E}}$. For a noticeable function $\epsilon(n) = n^{-O(1)}$, the extractor $\widehat{\mathsf{E}}$, is given as input a circuit C that $\left(\epsilon, \widehat{\mathcal{D}}\right)$-approximates f_{k} (where $\widehat{\mathcal{D}} = \widehat{\mathcal{D}}(1^n)$) and the parameter $1^{1/\epsilon}$. $\widehat{\mathsf{E}}$ first constructs from C a probabilistic circuit C' that, with probability $\left(1 - \frac{1}{\sqrt[4]{n}}\right)$, $((1 - \frac{1}{\sqrt[4]{n}}), \mathcal{D})$-approximates g_{k}. The circuit C', given a random sample $q \leftarrow \mathcal{D}(1^n)$, samples $(q'_1, \ldots, q'_n) \leftarrow \widehat{\mathcal{D}}(1^n)$ and replaces a random coordinate q'_{i^*} with q, and feeds the augmented n-fold sample to C. The circuit C' repeats this procedure $\frac{n}{\epsilon}$ times (for the same q), and if it obtains an answer $(z_1, \ldots, z_n)$ that is not $\perp$, it returns z_{i^*}. Then, after producing the circuit C', $\widehat{\mathsf{E}}$ then chooses n random strings $r_1, \ldots, r_n$ for C', and runs the extractor E of g_{k}, on each C'_{r_i}, if all executions output $\perp$, the extractor fails, and otherwise it produces k.

To show that $\widehat{\mathsf{E}}$ works as required, we show that the circuit C' indeed approximates g_{k} as required. First since C is assumed to err with negligible probability, so does C'; we now show that C' answers with high probability. For any sample $q \in \mathrm{supp}(\mathcal{D})$, let $\widehat{\mathcal{D}}|_q$ denote the distribution that the circuit C' samples from (i.e., where a random sample is drawn from $\widehat{\mathcal{D}}$, and then one of its random coordinates is replaced by q). We say that q is good if $\Pr_{\vec{q'} \leftarrow \widehat{\mathcal{D}}|_q}\left[C(\vec{q'}) \neq \perp\right] \geq \frac{\epsilon}{2}$. We claim that an input q drawn from $\mathcal{D}$ is good with probability at least $\left(1 - \frac{1}{\sqrt{n}}\right)$. Indeed, if that was not the case, then in any sample $\vec{q'} \leftarrow \widehat{\mathcal{D}}(1^n)$, there is some coordinate that is **not** good, with probability at least $\left(1 - \frac{1}{\sqrt{n}}\right)^n \approx e^{-\sqrt{n}}$; however, conditioned on this event, C answers a random sample with probability at most $\epsilon/2$, and thus overall it answers with probability at most $\epsilon(n)/2 - \mathrm{negl}(n) < \epsilon(n)$, leading to a contradiction. Now, since $\Pr_{q \leftarrow \mathcal{D}, C'}\left[C'(q) \neq \perp\right] \geq \left(1 - \frac{1}{\sqrt{n}}\right)$, it holds that for a $\left(1 - \frac{1}{\sqrt[4]{n}}\right)$-fraction of the random coins r used by C', it holds that $\Pr_{q \leftarrow \mathcal{D}}\left[C'_r(q) \neq \perp\right] \geq (1 - \frac{1}{\sqrt[4]{n}})$. Thus, one of the n repeated extraction attempts from C' using different randomness, will succeed, except with negligible probability. □

Further on, it will be convenient to also consider a notion where the circuit is required to approximate d distributions $\mathcal{D}_1, \ldots, \mathcal{D}_d$, instead of one (in this work, it will always be the case that d is a constant). This definition will only serve us as an intermediate step towards constructing families according to Definition 2.2, where $d = 1$. In particular, we will not require that each of these distribution can be sampled independently of k, but rather we will only require that, when i^* is chosen at random from $[d]$, it is possible to sample from $\mathcal{D}_{i^*}$, independently of k. As we shall see, the latter weak sampling guarantee (combined with parallel repetition) will be sufficiently powerful to construct an unobfuscatable function with a single distribution that is samplable independently of k.

We next present the definition. Since robustness can always be amplified, it will be sufficient and more convenient to describe the notion for a constant approximation factor, e.g., $1/2$.

Definition 2.4 (d-distribution robust unobfuscatable function)**.** *A family of functions $\mathcal{F}$ is a d-distribution $\frac{1}{2}$-robust unobfuscatable family with respect to an efficient relation $\mathcal{R}_{\mathcal{F}}$ and input samplers $\mathcal{D}_1, \ldots, \mathcal{D}_d$, if it is:*

1. **Black-box unlearnable:** *as in Definition 2.2.*
2. **Non-Black-box learnable:** *There exists an efficient extractor E such that for any large enough $n \in \mathbb{N}$, any $\mathsf{k} \in \{0,1\}^n$, and every circuit C that $(\frac{1}{2}, \mathcal{D}_i)$-approximates f_{k}, for all $i \in [d]$, E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from C:*

$$\Pr_{\mathsf{E}}\left[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n)\right] \geq 1 - \mathrm{negl}(n) \cdot \mathrm{poly}(|C|) \ ,$$

where $\mathcal{D}_i = \mathcal{D}_i(1^n, \mathsf{k})$, and poly *is a fixed polynomial depending only on E.*

For any $\mathsf{k} \in \{0,1\}^n$*, consider the distribution* $\mathcal{D}^*(\mathsf{k}) = \{q \leftarrow \mathcal{D}_{i^*}(1^n, \mathsf{k}) \mid i^* \leftarrow [d]\}$*. We say that* $\mathcal{D}_1, \ldots, \mathcal{D}_n$ *are* **jointly key-independent***, if there exists an efficient sampler* $\mathcal{D}^*$*, such that for all* k*,* $\mathcal{D}^*(1^n) \triangleq \mathcal{D}^*(\mathsf{k})$.

We now show that any d-distribution robust unobfuscatable family, where $\mathcal{D}_1, \ldots, \mathcal{D}_n$ are jointly key-independent, can be transformed in a robust unobfuscatable family according to Definition 2.2. Specifically, we show a $\frac{2}{3}$-robust family, but robustness can always be amplified using Lemma 2.1.

Lemma 2.2. *If* $\mathcal{G}$ *is a* d*-distribution* $\frac{1}{2}$*-robust unobfuscatable family, with samplers* $\mathcal{D}_1, \ldots, \mathcal{D}_d$ *that are jointly key-independent, then* $\mathcal{G}$ *can be transformed to a* $\frac{2}{3}$*-robustness* $\mathcal{F}$ *(with a single distribution).*

We now describe the construction behind the lemma, followed by the analysis.

Construction 2.2 ($\frac{2}{3}$-Robustness from d-distribution $\frac{1}{2}$-robustness). Let $\mathcal{G}$ be a d-distribution $\frac{1}{2}$-robust unobfuscatable family with samplers $\{\mathcal{D}_i\}_{i\in[d]}$. (Each $\mathcal{D}_i$ is possibly key-dependent, but $\{\mathcal{D}_i\}_{i\in[d]}$ are jointly key-independent). We construct a new robust unobfuscatable family $\mathcal{F}$ on a distribution ensemble $\widehat{\mathcal{D}}^*$ as follows. Each function $f_{\mathsf{k}} : \{0,1\}^{n^2} \to \{0,1\}^{n^2}$ is just the n-fold version $\widehat{g_{\mathsf{k}}}(q_1, \ldots, q_n) = (g_{\mathsf{k}}(q_1), \ldots, g_{\mathsf{k}}(q_n))$ of the function $g_{\mathsf{k}} : \{0,1\}^n \to \{0,1\}^n$. The input distribution $\widehat{D}^*$ is the n-fold $\mathcal{D}^* \times \cdots \times \mathcal{D}^*$, (where $\mathcal{D}^*$ is as defined above).

Proof sketch. Black-box unlearnability follows rather directly, from the fact that any oracle query to f_{k} can be perfectly simulated by n oracle queries to g_{k}. We thus focus on showing non-black-box learnability. First, note that since

Let C be a circuit that $\left(\frac{2}{3}, \widehat{\mathcal{D}}^*\right)$-approximates f_{k}, we construct a new probabilistic circuit $\widehat{C}'$ that, with overwhelming probability, for all $i \in [d]$, $(\frac{2}{3} - \frac{1}{\Omega(\sqrt{n})}, \mathcal{D}_i)$-approximates g_{k}. For this purpose, we describe another probabilistic circuit C' that, given a sample q, drawn from some $\mathcal{D}_j$, samples a random $i^* \in [d]$, and a random sample $(q'_1, \ldots, q'_n)$ from $\widehat{\mathcal{D}}^*$, replaces q'_{i^*} with q, and feeds it to C. The circuit $\widehat{C}'$, given a sample q, simply chooses n sets of random coins $r_1, \ldots, r_n$ for C', and runs each $C'_{r_i}(q)$; if any one of the circuits does not output $\perp$, $\widehat{C}'$ outputs the same. The extractor E for $\mathcal{F}$, then simply chooses random coins for $\widehat{C}'$ and feeds it to the extractor of g_{k}.

To show that the above works as required, we first show that, for any fixed $j \in [d]$, with $\left(\frac{1}{6} - \frac{1}{\Omega(\sqrt{n})}\right)$ probability, the circuit C' $\left(\frac{1}{2}, \mathcal{D}_j\right)$-approximates g_{k}. First, since C errs with negligible probability so does C'; let us show that C' does not output $\perp$ with the required probability. Let $\mathcal{D}'$ be the distribution that C' forwards to C (where sample i^* is replaced by a sample from $\mathcal{D}_j$). We can compute the statistical distance $\mathsf{SD}(\widehat{\mathcal{D}}^*, \mathcal{D}')$ by comparing the number $\#_j$ of samples taken from $\mathcal{D}_j$ in each one of them; a combinatoric calculation shows that this is bounded by:

$$\mathsf{SD}(\widehat{\mathcal{D}}^*, \mathcal{D}') \leq |\#_j(\widehat{\mathcal{D}}^*) - \#_j(\mathcal{D}')| \leq \sum_{k=0}^{n} \left| \mathsf{Bin}\left(k; n, \frac{1}{d}\right) - \mathsf{Bin}\left(k-1; n-1, \frac{1}{d}\right) \right| = O\left(\frac{d}{\sqrt{n}}\right) \ ,$$

where $\mathsf{Bin}(k; n, p) = \binom{n}{k} p^k (1-p)^{n-k}$ is the binomial probability with parameters (n, k, p). We shall assume for simplicity that $d = O(1)$, and our constructions indeed achieve this; the analysis extends also to any $d \ll \sqrt{n}$. It follows that C' obtains an answer with probability at least $\frac{2}{3} - \frac{1}{\Omega(\sqrt{n})}$, and in particular it holds with probability at least $\frac{1}{6} - \frac{1}{\Omega(\sqrt{n})}$ over the coins of C', that C' $\left(\frac{1}{2}, \mathcal{D}_j\right)$-approximates g_{k}. We call such random coins r of C' good for $\mathcal{D}_j$. It is left to note that the probability that $\widehat{C}'$ does not sample $r_1, \ldots, r_n$ that contain a some r that is good for all of the $\mathcal{D}_j$'s is bounded by $d \cdot \left(1 - \frac{1}{7}\right)^n$, from which it follows that with overwhelming probability C' gives the required approximation for all $\mathcal{D}_j$'s.

□

2.2 Robust Unobfuscatable Functions with a Hardcore Secret

The notion of robust (and error-robust) unobfuscatable function is defined with respect to an unlearnable relation $\mathcal{R}_{\mathcal{F}}$, where it is guaranteed that given black-box access to f_{k}, it is hard to fully learn some unlearnable secret $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$. We now present a natural variant of this definition, where each function has an associated hardcore secret that is indistinguishable from a random string, even given black-box access to the function; in contrast, the hardcore secret can be extracted from any circuit that approximates f_{k}. This variant will be useful for constructing unobfuscatable functions with a verifiable unlearnable secret (see Section 4), and for some applications to resettable protocol (see Section 5.5).

In the following definition $\mathcal{HC}_n = \left\{h : \{0,1\}^{\mathrm{poly}(n)} \to \{0,1\}^n\right\}$ will represent a family of functions (which we will call hardcore functions).

Definition 2.5 (Robust unobfuscatable functions with a hardcore secret)**.** *A family of functions $\mathcal{F} = \{f_{\mathsf{k}}\}_{\mathsf{k}\in\{0,1\}^n, n\in\mathbb{N}}$ is a robust unobfuscatable family with respect to an efficient input sampler $\mathcal{D}$, and a family of* **hardcore functions** *$\mathcal{HC}$ if it has the following properties:*

1. **Black-box indistinguishability:** *For any poly-size distinguisher $\mathsf{L} = \{\mathsf{L}_n\}_{n\in\mathbb{N}}$, and all large enough n:*

$$\left| \Pr_{(\mathsf{k},h)} \left[\mathsf{L}_n^{f_{\mathsf{k}}}(h, h(\mathsf{k})) = 1\right] - \Pr_{(\mathsf{k},h,u)} \left[\mathsf{L}_n^{f_{\mathsf{k}}}(h, u) = 1\right] \right| \leq \mathrm{negl}(n) \ ,$$

   where $\mathsf{k} \leftarrow \{0,1\}^n$, $h \leftarrow \mathcal{HC}_n$, and $u \leftarrow \{0,1\}^n$ are all sampled independently.

2. **Non-Black-box learnability:** *There exists an efficient extractor E such that for any noticeable function $\epsilon(n) = n^{-O(1)}$, any large enough n, any hardcore function $h \in \mathcal{HC}_n$, any $\mathsf{k} \in \{0,1\}^n$, and every circuit C that $(\epsilon, \mathcal{D})$-approximates f_{k}, $\mathsf{E}(h)$ extracts $h(\mathsf{k})$ from C:*

$$\Pr_{\mathsf{E}} \left[h(\mathsf{k}) \leftarrow \mathsf{E}(C, h, 1^n, 1^{1/\epsilon})\right] \geq 1 - \mathrm{negl}(n) \cdot \mathrm{poly}(|C|) \ .$$

Next, we show that any robust unobfuscatable family $\mathcal{G}$ that has a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$ (as defined in Remark 2.4) can be transformed into a robust $\mathcal{F}$ with a hardcore secret.

Lemma 2.3. *There exists a hardcore family $\mathcal{HC} = \{\mathcal{HC}_n\}_{n\in\mathbb{N}}$, such that a robust unobfuscatable family $\mathcal{G}$ with respect to a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$, can be transformed to a robust unobfuscatable family $\mathcal{F}$ with respect to some $\mathcal{R}_{\mathcal{F}}$ and the hardcore family $\mathcal{HC}$.*

Proof sketch. Given a robust family $\mathcal{G}$ with respect to a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$ and an efficient input sampler $\mathcal{D}$, we define a new function family $\mathcal{F}$, where each f_{k} consists of n independently chosen functions from $\mathcal{G}$, with keys $(\mathsf{k}'_1, \ldots, \mathsf{k}'_n)$; the new key k is set to be $(\mathsf{k}'_1, \ldots, \mathsf{k}'_n)$. The input sampler of the function is the n-fold product distribution $\widehat{\mathcal{D}} = \mathcal{D} \times \cdots \times \mathcal{D}$, and $h_{\mathsf{k}}(q_1, \ldots, q_n)$ is defined to be $g_{\mathsf{k}'_1}(q_1), \ldots, g_{\mathsf{k}'_n}(q_n)$.

Let us denote by $\ell = \ell(n)$ the length $\ell = |\mathsf{z}_i|$ of the unique witness $\mathsf{z}_i \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}'_i)$. The hardcore family $\mathcal{HC}_n$ will be the family $\mathcal{GL}_n$ that extracts a single Goldreich-Levin [GL89] hardcore-bit from each k'_i. That is, a randomly chosen function $h_{r_1,\ldots,r_n} \in \mathcal{GL}_n$ is parameterized by n random strings $\left\{r_i \in \{0,1\}^\ell\right\}$, and is defined as:

$$h_{r_1,\ldots,r_n}(\mathsf{k}'_1, \ldots, \mathsf{k}'_n) = \langle \mathsf{z}_1, r_1 \rangle, \ldots, \langle \mathsf{z}_n, r_n \rangle \ ,$$

where each $z_i \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}'_i)$ is the unique unlearnable secret corresponding to k'_i, and $\langle \cdot, \cdot \rangle$ is the inner-product operation modulo 2.

To see that $\mathcal{F}$ has the black-box indistinguishability property (Definition 2.5), note that the uniquely determined $\mathsf{z}_1, \ldots, \mathsf{z}_n$ are each uninvertible given oracle access to f_{k}; indeed, any inverting algorithm directly implies a learner for the underlying g_{k}. We can thus apply the Goldreich-Levin theorem [GL89] to deduce that $\langle \mathsf{z}_1, r_1 \rangle, \ldots, \langle \mathsf{z}_n, r_n \rangle$ are pseudo-random, given $r_1, \ldots, r_n$ and oracle access to f_{k}.

Next, we show that $\mathcal{F}$ is non-black-box learnable according to Definition 2.5; namely, the hardcore secret can be extracted from any circuit approximation. Indeed, note that any circuit C that $(\widehat{\mathcal{D}}, 2\epsilon)$-approximates f_{k} can be transformed, with overwhelming probability, into a circuit C_i that $(\mathcal{D}, \epsilon)$-approximates $g_{\mathsf{k}'_i}$ (and this holds for any $i \in [n]$). To transform C into such a circuit, we construct a circuit C' that, given a sample $q \leftarrow \mathcal{D}(1^n)$ for $g_{\mathsf{k}'_i}$, completes q into a sample from $\widehat{\mathcal{D}}$ by sampling the rest of the coordinates himself, and then feeds this tuple to C. A standard averaging argument shows that with probability at least ϵ over the choice of randomness r for C' the resulting circuit C'_r $(\mathcal{D}, \epsilon)$-approximates $g_{\mathsf{k}'_i}$. Thus we can take n/ϵ random copies of C', and with overwhelming probability get a circuit C_i that $(\mathcal{D}, \epsilon)$-approximates $g_{\mathsf{k}'_i}$. (It is important here that since C errs with negligible probability so does C' and hence we can do the above amplification.)

Thus the extractor $\mathsf{E}_{\mathcal{F}}$ for $\mathcal{F}$, would run the extractor $\mathsf{E}_{\mathcal{G}}$ with each circuit C_i, and obtain the corresponding unique secret $\mathsf{z}_i \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}'_i)$. In particular, given $h_{r_1,\ldots,r_n} \in \mathcal{GL}_n$, $\mathsf{E}_{\mathcal{G}}$ can compute as required

$$h_{r_1,\ldots,r_n}(\mathsf{k}) = h_{r_1,\ldots,r_n}(\mathsf{k}'_1, \ldots, \mathsf{k}'_n) = \langle \mathsf{z}_1, r_1 \rangle, \ldots, \langle \mathsf{z}_n, r_n \rangle \ .$$

This concludes the proof of Lemma 2.3. □

3 A Construction of Robust Unobfuscatable Functions

In this section, we construct robust unobfuscatable functions from one-way functions.

Theorem 3.1. *Assuming one-way functions, there exist a family of robust unobfuscatable functions.*

3.1 Required PRFs and Encryption

Before describing the construction, we define several required primitives.

We start by defining invoker randomizable pseudo-random functions. These are functions that allow their invoker to ensure that the output is truly uniform, independently of the seed for the function. Looking ahead, such functions will allow the extractor for the unobfuscatable function family, to obtain samples from the proper distribution. See further details in the next section. The definition is taken almost verbatim from [BGI$^+$01].

Definition 3.1 (Invoker randomizable pseudo-random functions [BGI$^+$01])**.** *Let* $\mathsf{PRF} = \{\mathsf{PRF}_{\mathsf{s}}\}_{\mathsf{s}\in\{0,1\}^*}$*, be pseudo random function family, where for* $\mathsf{s} \in \{0,1\}^n$*,* $\mathsf{PRF}_{\mathsf{s}} : \{0,1\}^{\ell(n)+n} \to \{0,1\}^n$*. Then* PRF *is called* **invoker randomizable** *if, for any* $\mathsf{s} \in \{0,1\}^n$ *and any* $x \in \{0,1\}^{\ell(n)}$*, the mapping* $r \mapsto \mathsf{PRF}_{\mathsf{s}}(x, r)$ *is a permutation.*

[BGI$^+$01] show that invoker randomizable PRFs are implied by any PRF.

The required symmetric-key encryption. In our construction, we will make use of a *symmetric-key encryption* scheme with specific properties. Next, we define these properties and note existing constructions that satisfy these properties, and based only on one-way functions. In a nutshell, we require a CCA-1 symmetric-key encryption scheme with public randomness, and oblivious generation of ciphers for a random plaintexts. Such an encryption scheme can be obtained using a one-bit output PRF as $\mathsf{Enc}_{\mathsf{sk}}(b) = (r, b \oplus \mathsf{PRF}_{\mathsf{sk}}(r))$.

In what follows, we give slightly more general definitions that are sufficient for our needs, and will be useful for optimization of extraction running time.

First we require that the encryption scheme is *decomposable*, meaning that every cypher can be "decomposed" into a *public part* and a *private part*. We require that the public part is independent of the plaintext, however, together with the plaintext, the public part uniquely defines the ciphertext (with respect to a given secret key). For example, any encryption that uses public randomness (i.e., the randomness of the encryption algorithm is included in the ciphertext) is decomposable; the public part of the cipher is just the public randomness.

Definition 3.2 (Decomposable encryption)**.** *A decomposable encryption scheme includes in addition to the standard* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is decomposable if there exist an efficient algorithm* pub *that operates on ciphertexts and satisfies the following conditions:*

- *For ciphertext* c*,* $\mathsf{pub}(\mathsf{c})$ *is independent of the plaintext and samplable; that is, there exist an efficient sampler* $\mathsf{PubSamp}$ *such that:*

$$\mathsf{PubSamp}(1^n) \triangleq \mathsf{pub}(\mathsf{Enc}_{\mathsf{sk}_n}(0)) \triangleq \mathsf{pub}(\mathsf{Enc}_{\mathsf{sk}_n}(1)) \ ,$$

  for any secret key $\mathsf{sk} \in \{0,1\}^n$ *(however,* $\mathsf{PubSamp}$ *works independently of* sk*).*

- *A ciphertext* c *is deterministically defined by* $\mathsf{pub}(\mathsf{c})$ *and the plaintext; that is, for every secret key* sk *and two distinct ciphers* $\mathsf{c} \neq \mathsf{c}'$*, if* $\mathsf{pub}(\mathsf{c}) = \mathsf{pub}(\mathsf{c}')$*, then* $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}) \neq \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}')$.

In addition, we shall require random generation of ciphertexts encrypting random values.

Definition 3.3 (Encryption with random ciphertext generation)**.** *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is said to have random ciphertext generation if there exist a ciphertext sampling algorithm* $\mathsf{RanSamp}$ *such that for any secret key* $\mathsf{sk} \in \{0,1\}^n$*:*

$$\mathsf{RanSamp}(1^n) \triangleq \mathsf{Enc}_{\mathsf{sk}_n}(U_1)) \ ,$$

were U_1 *is the uniform distribution over* $\{0,1\}$.

We remark that in both Definitions 3.2 and 3.3, the equality of distributions can be, naturally, replaced with statistical (or computational) indistinguishability; however, the construction presented below does satisfy the stronger notion with equality. We also remark that if we assume that the scheme has random ciphertext generation, then the algorithm $\mathsf{PubSamp}$, can be simply implemented as $\mathsf{PubSamp}(1^n) \triangleq \mathsf{pub}(\mathsf{RanSamp}(1^n))$, and need not be explicitly defined.

The encryption scheme used in our constructions. We will use a CCA-1 symmetric key decomposable bit encryption scheme with random ciphertext generation. Such an encryption scheme can be constructed from one-way functions. Concretely, given a PRF $\{f_{\mathsf{s}}\}_{\mathsf{s}\in\{0,1\}^*}$ with one bit output, for security parameter n, the secret key is a random $\mathsf{s} \in \{0,1\}^n$, and encryption of a bit b is computed by sampling a random $r \in \{0,1\}^n$ and outputting $r, f_{\mathsf{s}}(r) \oplus b$. This function can be shown to be CCA-1 (see [Gol00]), and it is clearly decomposable and has random cipher generation.

3.2 The Construction

To prove Theorem 3.1, we shall construct an unobfuscatable function family that is d-distribution $\frac{1}{2}$-robust with respect to distribution ensembles $\mathcal{D}_1, \ldots, \mathcal{D}_d$, that are jointly key-independent in the sense of Definition 2.4. Then, we will use Lemma 2.2 to obtain robust unobfuscatable functions (Definition 2.2). We now proceed to describe the construction (in several steps).

Construction 3.1. Let (a, b) be random strings, sk a key for a CCA-1 symmetric key decomposable encryption scheme with random ciphertext generation (Definitions 3.2,3.3), and let s a seed for a pseudorandom function PRF. The construction will use two probabilistic and three deterministic auxiliary functions:

1. The function $\mathbb{A}_{\mathsf{sk},a}$ (given whatever input) returns a bit encryption $\mathsf{c} = \mathsf{Enc}_{\mathsf{sk}}(a)$ of a's bits.

2. The function $\mathbb{H}_{\mathsf{sk}}$ is given two encryptions $\mathsf{c}_1, \mathsf{c}_2$, and an operation $\odot$ in a universal set of gates, it then decrypts to obtain the plaintexts x_1 and x_2, where $x_i = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}_i)$, it computes the operation: $x_3 = x_1 \odot x_2$, and returns an encryption $\mathsf{c}_3 = \mathsf{Enc}_{\mathsf{sk}}(x_3)$.

3. The function $\mathbb{R}_{\mathsf{s}}$ is given $r \in \{0,1\}^*$, and returns $\mathsf{PRF}_{\mathsf{s}}(r)$.

4. The function $\mathbb{R}_{a,b,\mathsf{s}}$ is given $r \in \{0,1\}^n$, and returns $b \oplus \mathsf{PRF}_{\mathsf{s}}(r \oplus a)$.

5. The function $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ is given n bit encryptions $\mathsf{c}_1, \ldots, \mathsf{c}_n$, it decrypts them to obtain plain text $r \in \{0,1\}^n$, and returns $b \oplus \mathsf{PRF}_{\mathsf{s}}(r \oplus b || \mathsf{pub}(\mathsf{c}_1) || \ldots || \mathsf{pub}(\mathsf{c}_n))$. (Recall that $\mathsf{pub}(\mathsf{c})$ is the public part of c.)

Making the functions deterministic and invoker randomizable. Let s' be a seed for an invoker randomizable PRF (Definition 3.1), we define derandomized variants $\mathbb{A}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}_{\mathsf{sk},\mathsf{s}'}$ of the two probabilistic functions: $\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}$. For a function $\mathbb{G}_{\mathsf{key}} \in \{\mathbb{H}_{\mathsf{sk}}, \mathbb{A}_{\mathsf{sk},a}\}$, the function $\mathbb{G}_{\mathsf{key},\mathsf{s}'}$ gets, in addition to its original input q, a random string $r \in \{0,1\}^n$. $\mathbb{G}_{\mathsf{key},\mathsf{s}'}(q;r)$ runs the original $\mathbb{G}_{\mathsf{key}}(q)$, using randomness $\mathsf{PRF}_{\mathsf{s}'}(q;r)$.

The n-fold repetition. For each function $\mathbb{G}$ above, we define an n-fold variant $\widehat{\mathbb{G}}$, that takes n inputs $q_1, \ldots, q_n$ and returns $(\mathbb{G}(q_1), \ldots, \mathbb{G}(q_n))$.

The function. A function $f_{\mathsf{k}} \in \mathcal{F}$ will be parameterized by a random $\mathsf{k} = (a, b, \mathsf{sk}, \mathsf{s}, \mathsf{s}') \in \{0,1\}^{5n}$, where $(a, b, \mathsf{sk}, \mathsf{s}, \mathsf{s}')$ are as specified above. The function f_{k} will be given input $(q, i_{\mathbb{G}})$ where $i_{\mathbb{G}} \in [5]$, indicates which function $\mathbb{G} \in \left\{\widehat{\mathbb{A}}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}, \widehat{\mathbb{R}}_{\mathsf{s}}, \widehat{\mathbb{R}}_{a,b,\mathsf{s}}, \widehat{\mathbb{R}}_{\mathsf{sk},b,\mathsf{s}}\right\}$ to invoke with input q. The unlearnable relation is

$$\mathcal{R}_{\mathcal{F}} = \left\{((a, b, \mathsf{sk}, \mathsf{s}, \mathsf{s}'), \tilde{b}) : b = \tilde{b}\right\} \quad .$$

(Note that this is a unique-witness relation according to Remark 2.4.

3.3 Black-Box Unlearnability

Lemma 3.1. *$\mathcal{F}$ given by Construction 3.1 satisfies black-box unlearnability with respect to $\mathcal{R}_{\mathcal{F}}$.*

We next prove the lemma. The high-level overview of the proof is presented in the introduction (Section 1.3).

Proof sketch. We shall perform the analysis in several steps: we will first show unlearnability assuming 1-fold probabilistic oracles then we will show security for the case that randomness is derived using (invoker-randomizable) PRF, and move to security for n-fold oracles, and finally for the actual function f_{k}.

Let $\mathcal{A}$ be a polysize adversary with oracle access to the functions $\mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$.

We first define E_1 to be the event that $\mathcal{A}$ produces distinct queries $q = (\mathsf{c}_1, \ldots, \mathsf{c}_n), q' = (\mathsf{c}'_1, \ldots, \mathsf{c}'_n)$ to $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ such that

$$r \oplus b || \mathsf{pub}(\mathsf{c}_1) || \ldots || \mathsf{pub}(\mathsf{c}_n) = r' \oplus b || \mathsf{pub}(\mathsf{c}'_1) || \ldots || \mathsf{pub}(\mathsf{c}'_n) \quad ,$$

where $(r, r') \in \{0,1\}^n$ are the decryptions under sk of (q, q').

Claim 3.1. $\Pr_{b,\mathsf{sk},\mathsf{s}}[E_1] = 0$.

Proof. Indeed, for E_1 to occur it must be that for all $i \in [n]$ $r_i = r'_i$ and $\mathsf{pub}(\mathsf{c}_i) = \mathsf{pub}(\mathsf{c}'_i)$, implying that $\mathsf{c}_i = \mathsf{c}'_i$, and thus also $q = q'$ (recall that the public part and the plaintext determine the cipher Definition 3.2). □

We now define E_2 to be the event that $\mathcal{A}$ produces queries $q = (\mathsf{c}_1, \ldots, \mathsf{c}_n)$ to $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ and q' to $\mathbb{R}_{\mathsf{s}}$ such that $r \oplus b || \mathsf{pub}(\mathsf{c}_1) || \ldots || \mathsf{pub}(\mathsf{c}_n) = q'$, where as above $r \in \{0,1\}^n$ is the underlying plaintext of q.

Claim 3.2. $\Pr_{b,\mathsf{sk},\mathsf{s}}[E_2] \leq \mathrm{negl}(n)$.

Proof. First, we note that the event is testable given (b, sk) and thus, it is enough to show that the claim holds when $\mathsf{PRF}_{\mathsf{s}}$, is replaced by a truly random function R.

Let $\bar{E}_i$ be the event that E_2 does **not** occur in the first i queries, it is enough to show that

$$\Pr\left[\bar{E}_{i+1} | \bar{E}_i\right] \geq 1 - \mathrm{negl}(n) \ .$$

Indeed, conditioned on $\bar{E}_i$, the view of $\mathcal{A}$ after the first i queries is information theoretically independent of b. We can now show that if $\bar{E}_{i+1}$ does not occur, then $\mathcal{A}$ can reconstruct b from its view in the first i queries, implying that E_{i+1} can only occur with negligible probability. Indeed, if the first $i+1$ queries contain q, q' satisfying E_2, then $b = r \oplus r'$, where r is the decryption of q and r' are the first n bits of q'. □

We now claim that an encryption schemes using sk is still semantically secure, even given the oracles $\mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$.

Claim 3.3 (semantic security)**.** *Let $\mathcal{A}$ be a polysize adversary, then:*

$$\left\{\mathcal{A}^{\mathbb{H}_{\mathsf{sk}},\mathbb{R}_{\mathsf{s}},\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}(\mathsf{Enc}_{\mathsf{sk}}(0))\right\}_{\mathsf{sk},\mathsf{s},b} \approx_c \left\{\mathcal{A}^{\mathbb{H}_{\mathsf{sk}},\mathbb{R}_{\mathsf{s}},\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}(\mathsf{Enc}_{\mathsf{sk}}(1))\right\}_{\mathsf{sk},\mathsf{s},b} \ ,$$

where the distributions are also over the randomness of all involved probabilistic functions.

Proof sketch. We first note that by Claims 3.2,3.1, we can replace $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ (in both distributions) with a PRF $\mathsf{PRF}_{\mathsf{s}'}$ that is completely independent of sk. Now, the claim follows directly from the CCA-1 security of the encryption scheme, just as in [BGI$^+$01, Claim 3.7]. □

Now, let $\mathcal{A}$ be a polysize adversary with oracle access to the (probabilistic) functions

$$\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}} \ .$$

We define E to be the event that $\mathcal{A}$ produces distinct queries (q, q') to $\mathbb{R}_{a,b,\mathsf{s}}$ and $\mathbb{R}_{\mathsf{s}}$, respectively, such that $q = q' \oplus a$.

Claim 3.4. $\Pr_{b,\mathsf{sk},\mathsf{s}}[E] = \mathrm{negl}(n)$.

Proof sketch. First we note that the event E is testable given a and thus, it is enough to show that the claim holds when $\mathsf{PRF}_{\mathsf{s}}$, is replaced always replaced by a truly random function R. Now assume towards contradiction that E occurs with noticeable probability $\epsilon = \epsilon(n)$. Then, there exists an $i \leq |\mathcal{A}|$ such that, the event E first occurs in the i'th query that $\mathcal{A}$ makes with noticeable probability $\epsilon/|\mathcal{A}|$. We consider an hybrid experiment, where for the first $i-1$ queries that $\mathcal{A}$ makes, $\mathbb{R}_{a,b,R}$ is replaced with an independent random function R'. The view of $\mathcal{A}$ does not change because the first i answers to $\mathbb{R}_{a,b,R}$ are uncorrelated to the answers of $\mathbb{R}_R$, since the queries are not a-correlated, and they're also uncorrelated to the answers of $\mathbb{R}_{\mathsf{sk},b,R}$, since the queries are of different length. Thus, in this hybrid experiment, E still occurs first in the i'th query with probability $\epsilon/|\mathcal{A}|$. Now, after replacing $\mathbb{R}_{a,b,R}$ with an independent random function R' for the first $i-1$ queries, we think about a new adversary $\mathcal{A}_i$ that

halts after making the i'th query. By semantic security Claim 3.3 (and a standard hybrid argument), we can replace the oracle $\mathbb{A}_{\mathsf{sk},a}$, with a new oracle $\mathbb{A}_{\mathsf{sk},0^n}$, while effecting the probability that E occurs in $\mathcal{A}_i$ only by a negligible amount. Now, the view of $\mathcal{A}_i$ is information theoretically independent of a, and thus the probability that it outputs to queries q, q', such that $a = q \oplus q'$ cannot be noticeable, leading to the required contradiction. □

Putting things together. Now, for any adversary $\mathcal{A}$ with oracle access to the functions

$$\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}} \ ,$$

we can first replace all applications of $\mathsf{PRF}_{\mathsf{s}}$ with a truly random function R; then, by Claim 3.4, we can replace $\mathbb{R}_{a,b,R}$ with an independent random function R'; Now, we can invoke semantic security (Claim 3.3) to replace $\mathbb{A}_{\mathsf{sk},a}$ with $\mathbb{A}_{\mathsf{sk},0^n}$, and finally we can invoke again Claims 3.2,3.1, to replace $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ with another independent random function R''. Overall, each change effects the probability that $\mathcal{A}$ outputs b, only with negligible probability; however, the view of $\mathcal{A}$ in the final hybrid is information theoretically independent of b, and thus, $\mathcal{A}$ cannot output b with noticeable probability.

Deducing unlearnability of f_{k}. Recall that the function f_{k} is implemented n-fold repetitions of the invoker randomizable deterministic variants of the above oracles. However, the view of an adversary interacting with f_{k} can be simulated from the above oracles. Indeed, first note that we can replace $\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}$ by their invoker-randomizable deterministic variants: every new call (y, r) to $\mathbb{G}_{\mathsf{key},\mathsf{s}'}$ is answered according to probabilistic oracle $\mathbb{G}_{\mathsf{key}}(y)$, and repeated calls are answered consistently. Then, we can replace any oracle $\mathbb{G}$ with its n-fold variant $\widehat{\mathbb{G}}$: every call to $\widehat{\mathbb{G}}$ is replaced by n calls to $\mathbb{G}$. □

3.4 Non-Black-Box Learnability

In this section, we prove that Construction 3.1 is non-black-box learnable in the sense of Definition 2.4. the high-level overview of the proof (namely the construction of an extractor), is given in Section 1.3.

The following notation will be useful in the proof below.

- We denote by $\mathbb{A}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(U)$ the distribution

  $$\left\{\mathbb{A}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(r) : r \leftarrow \{0,1\}^n\right\}$$

  for sampling a random cipher encrypting a's bits. We denote by $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(U)$ its n fold variant

  $$\left\{\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(\vec{r}) : \vec{r} \leftarrow \{0,1\}^{n\times n}\right\}$$

  .
- We denote by $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot;U)$ the distribution

  $$\left\{\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}\left((\mathsf{c},\mathsf{c}',\odot;r)\right) : r \leftarrow \{0,1\}^n\right\}$$

  for sampling a cipher encrypting the result of an homomorphic operation. We denote by $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot;U)$ the distribution

  $$\left\{\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}\left((\mathsf{c},\mathsf{c}',\odot;r_1),\ldots,(\mathsf{c},\mathsf{c}',\odot;r_n)\right) : \vec{r} \leftarrow \{0,1\}^n\right\}$$

  for sampling an n-fold cipher encrypting the result of the homomorphic computation.

We now describe the extraction procedure; we start by describing an extractor that is only required to work for circuit that perfectly compute a 1-fold version of each one of the underlying functions. Then, we will explain how to generalize this extractor for faulty circuits, in the n-fold case.

Construction 3.2. Let C be a circuit that perfectly implements a function $f_{\mathsf{k}} \in \mathcal{F}$. In particular, C consists of the following restricted circuits

$$\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}, \mathbb{R}^{\triangle}_{\mathsf{s}}, \mathbb{R}^{\triangle}_{a,b,\mathsf{s}}, \mathbb{R}^{\triangle}_{\mathsf{sk},b,\mathsf{s}} \ ,$$

perfectly implementing each of f_{k}'s underlying functions. The extractor E works according to the following steps:

1. Obtain a random encryption of a's bits using $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(U_n)$.
2. Sample a random $r \leftarrow \{0,1\}^n$; then, using the circuit $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\cdot; U_n)$, homomorphically compute

$$\begin{aligned}
\mathsf{c}_1 &= \mathsf{Enc}_{\mathsf{sk}}(\mathbb{R}^{\triangle}_{a,b,\mathsf{s}}(a \oplus r)) = \mathsf{Enc}_{\mathsf{sk}}(\mathsf{PRF}_{\mathsf{s}}(r) \oplus b) \ , \\
\mathsf{c}_2 &= \mathsf{Enc}_{\mathsf{sk}}(\mathbb{R}^{\triangle}_{\mathsf{s}}(r)) = \mathsf{Enc}_{\mathsf{sk}}(\mathsf{PRF}_{\mathsf{s}}(r)) \ , \\
\mathsf{c}_3 &= \mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}_1, \mathsf{c}_2, \oplus; U_n) = \mathsf{Enc}_{\mathsf{sk}}(b) \ .
\end{aligned}$$

3. Sample a random $r' \leftarrow \{0,1\}^n$; using the circuit $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\cdot; U_n)$, and cipher c_3 homomorphically compute $\mathsf{c}_4 = \mathsf{Enc}_{\mathsf{sk}}(b \oplus r')$, and now obtain b by computing:

$$\begin{aligned}
b \oplus \mathsf{PRF}_{\mathsf{s}}(r' || \mathsf{pub}(\mathsf{c}_4)) &= \mathbb{R}^{\triangle}_{\mathsf{sk},b,\mathsf{s}}(\mathsf{c}_4) \\
\mathsf{PRF}_{\mathsf{s}}(r' || \mathsf{pub}(\mathsf{c}_4)) &= \mathbb{R}^{\triangle}_{\mathsf{s}}(r' || \mathsf{pub}(\mathsf{c}_4)) \ .
\end{aligned}$$

A "parallel emulation" of the extractor. Next, we move to describe an extractor $\widehat{\mathsf{E}}$ that extracts b from any circuit C that approximately implements the n-fold variants of the underlying functions related to $f_{\mathsf{k}} \in \mathcal{F}$. (The required notion of approximation is described following the construction, and is not needed for the description of the extractor, but only for its analysis.) At a very high-level, in the n-fold version of the extractors, a ciphertext for any single bit b is, in fact, an n-fold cipher consisting of n ciphers, each encrypting the same bit b.

Construction 3.3. The extractor $\widehat{\mathsf{E}}$ is given a circuit C consisting of restricted circuits

$$\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}, \widehat{\mathbb{R}}^{\triangle}_{\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{a,b,\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{\mathsf{sk},b,\mathsf{s}} \ ,$$

approximating each of f_{k}'s underlying n-fold functions. The extractor $\widehat{\mathsf{E}}$ runs a "parallel emulation" of E as follows.

Emulating ciphertext operations. We now describe how to perform ciphertext operations; specifically, how to retrieve encryptions of a, and how to perform homomorphic operations.

- Whenever E would run the circuit $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}$ to obtain an encryption of a's bits, $\widehat{\mathsf{E}}$ obtains n samples from $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(U)$. If all samples are $\perp$ abort; otherwise, continue with the first n-fold cipher that was successfully sampled.
- Whenever E would run the circuit $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ with two ciphers $(\mathsf{c}, \mathsf{c}')$ and a gate $\odot$, $\widehat{\mathsf{E}}$ has at hand two n-fold ciphers $\mathsf{c}_1, \ldots, \mathsf{c}_n$ and $\mathsf{c}'_1, \ldots, \mathsf{c}'_n$.
  For each pair $(\mathsf{c}_i, \mathsf{c}'_i)$: it obtains n samples from $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}_i, \mathsf{c}'_i, \odot; U)$. If at least a $\frac{1}{8}$-fraction of the samples are not $\perp$, exit the loop and continue with the first n-fold cipher as the result of the homomorphic operation; otherwise, continue to the next pair of ciphers. If non of the pairs produced sufficiently many samples, abort.

Emulating Steps 2 and 3 of E. Before describing how to perform a parallel emulation of Steps 2 and 3, we describe several auxiliary functions that will be used by $\widehat{\mathsf{E}}$.

For $\mathbb{G} \in \{\mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{b,\mathsf{sk},\mathsf{s}}\}$, let $\mathsf{RandInput}_{\mathbb{G}}$ be a function that samples a random input for $\mathbb{G}$:

- $\mathsf{RandInput}_{\mathbb{R}_\mathsf{s}}$ and $\mathsf{RandInput}_{\mathbb{R}_{a,b,\mathsf{s}}}$, each consist of n random strings $\vec{r} = r_1, \ldots, r_n$. (We note that $\mathbb{R}_\mathsf{s}$ may get different input lengths; throughout, the proper length will be clear from the context.)
- $\mathsf{RandInput}_{\mathbb{R}_{b,\mathsf{sk},\mathsf{s}}}$ consists of n random ciphers of random bits $\vec{\mathsf{c}} = \mathsf{c}_1, \ldots, \mathsf{c}_n$, sampled using $\mathsf{RanSamp}(1^n)$, as defined in 3.3.

Importantly, all the above samplers work **independently of the key** k for the unobfuscatable function, so E can invoke them.

For $\mathbb{G} \in \{\mathbb{R}_\mathsf{s}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{b,\mathsf{sk},\mathsf{s}}\}$, we define an auxiliary function $\mathsf{ParEval}_\mathbb{G}$ that, given queries $q_1, \ldots, q_n$ for $\mathbb{G}$, plants each q_i in a random n-fold query, and tries to obtain an answer. Intuitively, this function exploits the parallel repetition, to boost up the answering probability on any specific query q. The function is given by Algorithm (3.1).

Algorithm 3.1 Parallel Evaluation - $\mathsf{ParEval}_\mathbb{G}$

Input: $(q_1, \ldots, q_n)$

for $j \in [n]$ **do**
  obtain $(q'_1, \ldots, q'_n) \leftarrow \mathsf{RandInput}_\mathbb{G}$
  sample a random $i^* \leftarrow [n]$
  execute $\widehat{\mathbb{G}}(q'_1, \ldots, q'_{i^*-1}, q_j, q'_{i^*+1}, \ldots, q'_n)$
  if the call succeeds (does not output $\perp$) **then**
    obtain the answer $(a_1, \ldots, a_n)$ and set $\mathsf{ans}_j = a_{i^*}$
  else
    set $\mathsf{ans}_j = \perp$
  end if
end for
return $(\mathsf{ans}_1, \ldots, \mathsf{ans}_n)$

Emulating Step 2 of E. In this step, $\widehat{\mathsf{E}}$ has an encryption of a and it transforms it to an encryption of b, by homomorphically evaluating a circuit $C_{a,b}$ that outputs b on input a. The circuit $C_{a,b}$ will be constructed by $\widehat{\mathsf{E}}$ and will emulate a parallel execution of Step 2 of E. The circuit is described in Algorithm (3.2). We shall describe $C_{a,b}$ as a probabilistic circuit (using probabilistic subroutines); when $\widehat{\mathsf{E}}$ homomorphically computes $C_{a,b}$, it first samples the required random coins and hardwires them to $C_{a,b}$.

Algorithm 3.2 The a to b circuit - $C_{a,b}$

Input: $\tilde{a}$ (allegedly $\tilde{a} = a$)

sample $r_1, \ldots, r_n \in \{0,1\}^n$
execute $(\mathsf{ans}_1, \ldots, \mathsf{ans}_n) \leftarrow \mathsf{ParEval}_{\mathbb{R}_\mathsf{s}}(r_1, \ldots, r_n)$
execute $(\mathsf{ans}'_1, \ldots, \mathsf{ans}'_n) \leftarrow \mathsf{ParEval}_{\mathbb{R}_{a,b,\mathsf{s}}}(\tilde{a} \oplus r_1, \ldots, \tilde{a} \oplus r_n)$
if $\exists j \in [n]$ such that ans_j and ans'_j are not $\perp$ **then**
  return $\mathsf{ans}_j \oplus \mathsf{ans}'_j$
else
  return $\perp$
end if

Emulating Step 3 of E. In this step, $\widehat{\mathsf{E}}$ has an encryption of b, which it will transform into b. The above is done using the procedure C_b, given by Algorithm (3.3):

Algorithm 3.3 The b decoding procedure - C_b

Input: n tuples $\vec{\mathsf{c}}_1, \ldots, \vec{\mathsf{c}}_n$, each $\vec{\mathsf{c}}_i$ consists of n ciphers with an underlying plaintext $\tilde{b}_i$ (allegedly $\tilde{b}_i = b_i$).

sample $r_1, \ldots, r_n \in \{0,1\}^n$
for $j \in [n]$ **do**
  homomorphically compute from $\vec{\mathsf{c}}_1, \ldots, \vec{\mathsf{c}}_n$ new tuples of ciphers $\vec{\mathsf{c}'}_1, \ldots, \vec{\mathsf{c}'}_n$, encrypting $r'_j = \tilde{b} \oplus r_j$
  set $\vec{\mathsf{c}}^*_k = \vec{\mathsf{c}'}_1[k], \ldots, \vec{\mathsf{c}'}_n[k]$ to be an encryption of the bits of r'_j
  execute $(\mathsf{ans}_1, \ldots, \mathsf{ans}_n) \leftarrow \mathsf{ParEval}_{\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}(\vec{\mathsf{c}}^*_1, \ldots, \vec{\mathsf{c}}^*_n)$
  execute $(\mathsf{ans}'_1, \ldots, \mathsf{ans}'_n) \leftarrow \mathsf{ParEval}_{\mathbb{R}_\mathsf{s}}(r_j || \mathsf{pub}(\vec{\mathsf{c}}^*_1), \ldots, r_j || \mathsf{pub}(\vec{\mathsf{c}}^*_n))$
  if $\exists m \in [n]$ such that ans_m and ans'_m are not $\perp$ **then**
    return $\mathsf{ans}_m \oplus \mathsf{ans}'_m$
  end if
end for
return $\perp$ (if the loop failed for all $j \in [n]$)

Emulating all steps together. We describe the full extraction process of $\widehat{\mathsf{E}}$. $\widehat{\mathsf{E}}$ first constructs the circuit $C_{a,b}$ and the procedure C_b from the restricted circuits $(\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}, \widehat{\mathbb{R}}^{\triangle}_{\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{a,b,\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{\mathsf{sk},b,\mathsf{s}})$ (in particular, it will sample random coins to be used by $C_{a,b}$ and C_b). $\widehat{\mathsf{E}}$ then proceeds according to the following steps:

1. Obtain a random encryption of a's bits using $\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(U_n)$.
2. Using homomorphic ciphertext operations, evaluate the circuit $C_{a,b}$ on the encryptions of a's bits and obtain encryptions of b's bits.
3. Execute C_b on the encryptions of b's bits, obtain b and output it.

The approximated distributions. We define $d = O(1)$ distribution ensembles (or samplers) $\widehat{\mathcal{D}}_1, \ldots, \widehat{\mathcal{D}}_d$. For the extractor $\widehat{\mathsf{E}}$ to work, the circuit will be required to $(\widehat{\mathcal{D}}_i, \epsilon)$-approximate each of these distributions. Each $\widehat{\mathcal{D}}_i = \widehat{\mathcal{D}}_i(1^n)$ will be an n-fold version of a distribution $\mathcal{D}_i = \mathcal{D}_i(1^n)$; we thus define the distributions $\{\mathcal{D}_i\}$:

- $\mathcal{D}_{\mathbb{A}_{\mathsf{sk},a}}$ consists of n random strings $r_1, \ldots, r_n \in \{0,1\}^n$ (each r_i is "invoker randomness" for encrypting a_i).
- For each operation $\odot$ (from the universal set of gates) and every pair of bits x, x', $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_\mathsf{sk}}$ consists of a random string $r \in \{0,1\}^n$ (for encrypting $x \odot x'$), and random encryptions c, c' under sk with underlying plaintexts x, x'.
- $\mathcal{D}^n_{\mathbb{R}_\mathsf{s}}$ consists of a random string $r \in \{0,1\}^n$.
- $\mathcal{D}^{n+n^2}_{\mathbb{R}_\mathsf{s}}$ consists of a random string $r \in \{0,1\}^n$, and n independent samples from $\mathsf{PubSamp}(1^n)$ of random "public parts of a cipher" as defined in 3.2.
- $\mathcal{D}_{\mathbb{R}_{a,b,\mathsf{s}}}$ consists of a random string $r \in \{0,1\}^n$.
- $\mathcal{D}_{\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}$ consists n random encryptions $\mathsf{c}_1, \ldots, \mathsf{c}_n$ to the bits of a random string $r \in \{0,1\}^n$.

As noted above, for each $\mathcal{D}$ we define the n-fold version $\widehat{\mathcal{D}}$. For all the distributions $\mathcal{D}$, but $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_\mathsf{sk}}$, $\widehat{\mathcal{D}}$ is just the n-fold product distribution $\mathcal{D} \times \cdots \times \mathcal{D}$. For $\mathcal{D} = \mathcal{D}^{\odot,x,x'}_{\mathbb{H}_\mathsf{sk}}$, $\widehat{\mathcal{D}}$ is defined a bit differently: it is the product distribution $\mathcal{D} \times \ldots, \times \mathcal{D}$ **conditioned** on c, c' being the same in all repetitions; that is, the distribution corresponds to a single pair c, c', and n random strings $r_1, \ldots, r_n$ meant for n independent homomorphic operations. In addition, each of the distributions $\widehat{\mathcal{D}}$ above also includes an index $i_\mathbb{G} \in [5]$, indicating which function $\mathbb{G} \in \left\{\widehat{\mathbb{A}}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}, \widehat{\mathbb{R}}_\mathsf{s}, \widehat{\mathbb{R}}_{a,b,\mathsf{s}}, \widehat{\mathbb{R}}_{\mathsf{sk},b,\mathsf{s}}\right\}$ to invoke.

Key-independent sampling. Note that in Definition 2.4 for d-distribution robustness, we do not require that all the distributions $\mathcal{D}_1, \ldots, \mathcal{D}_d$ are efficiently samplable independently of the key k for the unobfuscatable function. Indeed, while almost all the above distributions can be sampled independently

of k, the distributions of the form $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_{\mathsf{sk}}}$ may not be samplable without the secret encryption key sk, even given (x, x'); we only assume sampling of ciphers for random plaintexts (Definition 3.3). We do require, however, that the distributions $\{\mathcal{D}_i\}$ are jointly key-independent (Definition 2.4), implying that, for a random $i^* \leftarrow [d]$, we can sample from $\mathcal{D}_{i^*}$, independently of k. (This sampler was denote by $\mathcal{D}^*$.) In our case, this is satisfied due to random ciphertext generation (Definition 3.3): instead of sampling at random $x, x' \in \{0,1\}$, and then sampling encryptions $(\mathsf{c}, \mathsf{c}')$ of (x, x'), we can directly sample two independent ciphers of random bits.

Furthermore, the extractor $\widehat{\mathsf{E}}$ itself works by generating query samples from the different distributions $\mathcal{D}_i$. However, $\widehat{\mathsf{E}}$ naturally cannot use the key k to sample (it is trying to extract it, or a part of it). For those distributions that can be sampled without k this is not a problem. On the other hand, for the distributions $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_{\mathsf{sk}}}$, where the secret key is needed, $\widehat{\mathsf{E}}$ does not generate the samples on its own, but rather they are generated as answers to previous ciphertext queries (to $\mathbb{H}$, or $\mathbb{A}$), without $\widehat{\mathsf{E}}$ knowing the underlying plaintext. The only place where $\widehat{\mathsf{E}}$ needs to generate ciphers completely on its own is when running $\mathsf{ParEval}_{\mathbb{R}_{b,\mathsf{sk},\mathsf{s}}}$ within C_b; however, there it only generates ciphers of random bits, which can already be done obliviously of the key for the encryption sk.

Remark 3.1 (One uniform distribution). In the general Definition 2.2 of unobfuscatable functions, we allow the distribution ensemble $\mathcal{D}$ to be an arbitrary samplable distribution. As mentioned in Remark 2.2, we can consider a more strict (but natural) definition where $\mathcal{D}$ is required to be the uniform distribution. Our constructions can indeed achieve this stronger notion. Indeed, the distribution $\mathcal{D}$ in our eventual construction of robust robust unobfuscatable functions, will be a n-fold repetition of the distribution $\mathcal{D}^*$, and $\mathcal{D}^*$ can be made uniform.

Specifically, using the symmetric key encryption described in Section 3.1, the samplers $\mathsf{PubSamp}$ and $\mathsf{RanSamp}$ both output uniformly random strings. So each distribution $\widehat{\mathcal{D}}_j$ described above simply consists of a random strings (of some length) and an index $i_{\mathbb{G}} \in [5]$. In particular, by appropriately padding with extra randomness, we can think of each of the n blocks of $\mathcal{D}^*$ as a uniform string representing a random $i_{\mathbb{G}} \in [5]$, plus an extra random string representing the input to $\mathbb{G}$. (There are some extra technical issues that can be easily taken care of: (a) the index $i_{\mathbb{H}}$ is outputted by several distributions $\widehat{\mathcal{D}}_j$, and not just one, which may skew the uniformity of $i_{\mathbb{G}}$; however, this can be taken care of by artificially adding more indices to represent $i_{\mathbb{H}}$. (b) formally, the way we defined things $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_{\mathsf{sk}}}$ includes n appearances of the same pair $(\mathsf{c}, \mathsf{c}')$; however, this was done just to be consistent with the description of $\widehat{\mathbb{H}}$ as a n-fold function, and can be augmented to include one copy.)

We now move on to proving that the construction described in this section is non-black-box learnable.

Lemma 3.2. *$\mathcal{F}$ given by Construction 3.1 is non-black-box learnable in the sense of Definition 2.4.*

Proof. We start by giving a roadmap to the proof; the high-level ideas behind the proofs are described in the introduction.

Overview of the proof. Our first step is to show the completeness homomorphic evaluation; namely, that when the extractor performs homomorphic operations (in its parallel mode), it will never get stuck, and will always obtain a new n-fold cipher representing the result of the homomorphic computation. At the second step, we will prove that the circuit $C_{a,b}$ constructed by the extractor indeed performs properly; that is, maps a to b. Finally, we will show that the procedure C_b, when given the n-fold encryption of b produced by $\widehat{\mathsf{E}}$, successfully outputs b.

Throughout the analysis, we will condition on the event that the circuit C (that we extract from) does not make any undetectable errors (i.e., $C(q) \notin \{f_{\mathsf{k}}(q), \perp\}$) on any query q sampled by the extractor for one of the functions implemented by the circuit. Recall, that the probability of an undetectable error is $\mathrm{negl}(n)$, and the total number of queries made by the extractor is bounded by some $\mathrm{poly}(|C|, n)$. Thus, this condition is violated and may cause extraction failure with probability at most $\mathrm{negl}(n) \cdot \mathrm{poly}(|C|)$.

Completeness of the homomorphic evaluation phase. We say that a pair of ciphers $(\mathsf{c}, \mathsf{c}')$ encrypting a pair bits x and x' is good for $\odot$ if

$$\Pr\left[\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}(\mathsf{c}, \mathsf{c}', \odot; U)\right] \geq \frac{1}{4} \ ,$$

where $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}(\mathsf{c}, \mathsf{c}', \odot; U)$ is as defined in Construction 3.3. We say that a pair of n-fold ciphers $\vec{\mathsf{c}}, \vec{\mathsf{c}'}$, each encrypting a pair of bits x and x', respectively, is good for $\odot$ if there exists $i \in [n]$ such that $(\vec{\mathsf{c}}[i], \vec{\mathsf{c}'}[i])$ is good for $\odot$. We say that a pair of n-fold ciphers is good if it is good for any operation $\odot$.

Since the circuit $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}$ answers $\mathcal{D}^{\odot,x,x'}_{\mathbb{H}_{\mathsf{sk}}}$, where c, c' are random, with probability $\frac{1}{2}$, it holds that at least an $\frac{1}{4}$-fraction of the cipher-pairs encrypting x, x' is good for $\odot$. Therefore, a random pair of n-fold ciphers is **not** good with probability at most $\left(\frac{3}{4}\right)^n \cdot d$ (indeed, the number of distributions d is a bound on the number of operations).

Claim 3.5. *The extractor $\widehat{\mathsf{E}}$ does not abort during the homomorphic evaluation phase, except with negligible probability.*

Proof sketch. We first claim that except with negligible probability, for every two n-fold ciphers $\vec{\mathsf{c}}$ and $\vec{\mathsf{c}'}$ that are produced at some point during the execution of $\widehat{\mathsf{E}}$ (at different times), the pair $(\vec{\mathsf{c}}, \vec{\mathsf{c}'})$ is good. Indeed, note that any two executions of circuit $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}$ produce a pair $(\vec{\mathsf{c}}, \vec{\mathsf{c}'})$ that is not good, only if the corresponding two executions of the function $\widehat{\mathbb{H}}_{\mathsf{sk}}$ (that never outputs $\perp$), given the same inputs, output such a pair. However, since $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}$ is invoker randomizable, the output of any two executions is uniformly distributed over all pairs of n-fold ciphers, and, therefore, is not good only with negligible probability $\left(\frac{3}{4}\right)^n \cdot d$; in particular, this is also the case for any pair produced by an execution of the circuit $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}$. The same argument holds for pair of n-fold ciphers generated by the circuit $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk}}$ (or by mixed pairs created by the two circuits).

We now claim that, except with negligible probability, the extractor does not abort during the homomorphic evaluation phase. We condition on the (overwhelmingly often event) that the extractor only runs a homomorphic operation only for good pairs. First note that, in Step 3.3, $\widehat{\mathsf{E}}$ aborts only if all n samples from $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(U)$ are $\perp$. Thus, because the circuit $\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a}$ answers on $\widehat{\mathcal{D}}_{\mathbb{A}_{\mathsf{sk},a}}$ with probability $\frac{1}{2}$, $\widehat{\mathsf{E}}$ aborts only with probability 2^{-n}. In Step 3.3, when $\widehat{\mathsf{E}}$ is running a homomorphic operation $\odot$ for a good pair of n-fold ciphers $(\vec{\mathsf{c}}, \vec{\mathsf{c}'})$, there always exist some $i \in [n]$, such that the pair of ciphers $(\vec{\mathsf{c}}[i], \vec{\mathsf{c}'}[i])$ is good for the $\odot$. Therefore, samples drawn from $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\vec{\mathsf{c}}[i], \vec{\mathsf{c}'}[i], \odot; U)$ are not $\perp$ with probability at least $\frac{1}{4}$. Recall, that $\widehat{\mathsf{E}}$ aborts only if it fails to obtain more than $\frac{1}{8}$-fraction of the samples in all iterations; however, this would occur in the good iteration i only with negligible probability. □

Good inputs under completion. We next analyze the completeness of $C_{a,b}$, and C_b. For this purpose, we prove a simple claim regarding the probability of successfully computing a function on a specific sample taken from a 1-fold distribution. We will compute the function by completing the specific sample into a random sample and evaluate the function on it.

For $\mathbb{G} \in \{\mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}\}$, let $\widehat{\mathcal{D}}_{\mathbb{G}}$ be an n-fold samplable distribution ($\widehat{\mathcal{D}}_{\mathbb{G}} = \mathcal{D}_{\mathbb{G}} \times \cdots \times \mathcal{D}_{\mathbb{G}}$). For a fixed element q in the support of $\mathcal{D}_{\mathbb{G}}$, we denote by $\widehat{\mathcal{D}}_{\mathbb{G}}|_q$ the distribution given by sampling a random $\vec{q'}$ from $\widehat{\mathcal{D}}_{\mathbb{G}}$ and replacing a random coordinate q'_{i_*} with q. We say that q is good for $\mathcal{D}_{\mathbb{G}}$ if

$$\Pr_{\vec{q'} \leftarrow \widehat{\mathcal{D}}_{\mathbb{G}}|_q}\left[\mathbb{G}(\vec{q'}) \neq \perp\right] \geq \frac{1}{4} \ .$$

Claim 3.6. $\Pr_{q \leftarrow \mathcal{D}_{\mathbb{G}}}[q \textit{ is good for } \mathcal{D}_{\mathbb{G}}] \geq 1 - \frac{1}{\sqrt{n}}$.

Proof sketch. First recall that for $\mathbb{G} \in \{\mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}\}$, it holds that

$$\Pr_{\vec{q'} \leftarrow \widehat{\mathcal{D}}_{\mathbb{G}}} \left[\mathbb{G}(\vec{q'}) \neq \perp \right] \geq \frac{1}{2} \ .$$

Assume towards contradiction that the claim does not hold, then except with negligible probability $\left(1 - \frac{1}{\sqrt{n}}\right)^n \approx e^{-\sqrt{n}}$, a random sample from $\mathcal{D}$ contains some bad q; however, conditioned on this event, the event $\left\{ \mathbb{G}(\vec{q'}) \neq \perp \right\}$ occurs with probability at most $\frac{1}{4}$, resulting in a contradiction. □

Completeness of the circuit $C_{a,b}$. We now show that the probabilistic circuit $C_{a,b}$ constructed by the extractor (almost always) returns b on input a as required.

Claim 3.7. $\Pr_{C_{a,b}}[C_{a,b}(a) = b] \geq 1 - \mathrm{negl}(n)$.

Proof sketch. Recall that $C_{a,b}$ samples n random string $r_1, \ldots, r_n$. Conditioned on r_i being good for $\mathcal{D}_{\mathbb{R}_{\mathsf{s}}}$, $\mathsf{ans}_i \neq \perp$ with probability at least $\frac{1}{4}$; similarly, if $r_i \oplus a$ is good for $\mathcal{D}_{\mathbb{R}_{a,b,\mathsf{s}}}$, $\mathsf{ans}'_i \neq \perp$ with probability at least $\frac{1}{4}$. By Claim 3.6, for a random r_i, the pair $(r_i, r_i \oplus a)$ is simultaneously good (in the above sense) with probability at least $1 - \frac{2}{\sqrt{n}}$, in which case both ans_i and ans'_i are not $\perp$ and $b = \mathsf{ans}_i \oplus \mathsf{ans}'_i$, with probability at least $\frac{1}{16}$; overall, b is returned except with probability $\left(\frac{1}{8\sqrt{n}}\right)^n$. □

Completeness of the procedure C_b. We now show that the procedure C_b (almost always) returns b on an n-fold encryption of b as required.

Claim 3.8. *Let $\vec{\mathsf{c}}$ be the n-fold encryption of b produced by $\widehat{\mathsf{E}}$, then* $\Pr_{C_b}[C_b(\vec{\mathsf{c}}) = b] \geq 1 - \mathrm{negl}(n)$.

Proof sketch. Throughout, we condition on the (overwhelmingly often) event that homomorphic operations do not fail. Let c be a (1-fold) random encryption of a random n-bit string r; by Claim 3.6, c is good for $\mathcal{D} = \widehat{\mathcal{D}}_{\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}$ with probability at least $1 - \frac{1}{\sqrt{n}}$. For such a random c, let $r(\mathsf{c}) = \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c})$ be the underlying plaintext, and let $\mathsf{pub}(\mathsf{c})$ be the public part of c. We say that c is good for $\mathcal{D} = \widehat{\mathcal{D}}^{n+n^2}_{\mathbb{R}_{\mathsf{s}}}$, if $(b \oplus r(\mathsf{c}), \mathsf{pub}(\mathsf{c}))$ is good for $\mathcal{D} = \widehat{\mathcal{D}}^{n+n^2}_{\mathbb{R}_{\mathsf{s}}}$. Note that a random cipher of a random string, such as c, induces a random pair $(b \oplus r(\mathsf{c}), \mathsf{pub}(\mathsf{c}))$; thus, by Claim 3.6 c is good for $\mathcal{D} = \widehat{\mathcal{D}}^{n+n^2}_{\mathbb{R}_{\mathsf{s}}}$ with probability at least $1 - \frac{1}{\sqrt{n}}$. Overall, a random c is good for both distributions with probability at least $1 - \frac{2}{\sqrt{n}}$. We say that r is good if a random encryption c of r is good for both distributions with probability $1 - \sqrt[4]{\frac{4}{n}}$; in particular, we know that there is a $\left(1 - \sqrt[4]{\frac{4}{n}}\right)$-fraction of good r's.

For a fixed r, let M be a matrix with rows $(\vec{\mathsf{c}}_1, \ldots, \vec{\mathsf{c}}_n)$, such that each $\vec{\mathsf{c}}_i$ is an n-fold cipher encrypting the bit r_i. We say that such a matrix M is good if one of its columns $M[j] = (\vec{\mathsf{c}_1}[j], \ldots, \vec{\mathsf{c}}_n[j])$ is a good (1-fold) encryption of the string r (as defined above). By the previous paragraph, it follows that if r is good, then a random matrix M encrypting it is **not** good with probability at most $\left(\sqrt[4]{\frac{4}{n}}\right)^n \leq \left(\frac{1}{17}\right)^n$.

Recall that, for random $r_1, \ldots, r_n$ C_b generates for each $r = r_j$ a matrix M encrypting $r' = b \oplus r$, where every row of M is an n-fold cipher encrypting r'. Fix any column $M[i]$ of the matrix, this column is generated homomorphically from a pair of ciphers $(\mathsf{c}, \mathsf{c}')$. We claim that except with negligible probability $\Pr\left[\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}, \mathsf{c}', \odot; U) \neq \perp\right] \geq \frac{1}{16}$; otherwise, the homomorphic evaluation procedure (Step 3.3) would skip $(\mathsf{c}, \mathsf{c}')$ with overwhelming probability. Next, note that the output of the (non-aborting) invoker-randomizable function $\widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}, \mathsf{c}', \odot; U)$ is uniformly distributed over all n-fold ciphers of r'_i. It follows that $M[i]$ that is sampled from the circuit $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}, \mathsf{c}', \odot; U)$ is uniformly distributed over at least a $\frac{1}{16}$-fraction of all n-fold cipher encrypting r'_i. Since this holds for every i, the matrix M is uniformly distributed over a $\left(\frac{1}{16}\right)^n$-fraction of all random cipher matrices encrypting r'.

It now follows, that for any good r, a matrix M encrypting r that is produced by C_b is good except with probability $\left(\frac{16}{17}\right)^n$. For any good M there exists a column c_k that is a good encryption of r, and in particular ans_k and ans'_k are not $\perp$, and $b = \mathsf{ans}_k \oplus \mathsf{ans}'_k$ with probability $\frac{1}{16}$ in each one of the n trials. Finally, since a $\left(1 - \sqrt[4]{\frac{4}{n}}\right)$-fraction of the r's are good, and our extractor performs n trials, C_b will recover b with overwhelming probability.

□

This concludes the proof of Lemma 3.2.

□

3.4.1 Necessity of One-Way Functions

We show that one-way functions are not only sufficient for constructing robust unobfuscatable functions, but also necessary (given the natural requirement of a unique witness unlearnable relation noted in Remark 2.4).

Lemma 3.3. *Robust unobfuscatable functions with unique witness unlearnable relations imply one-way functions.*

Proof sketch. Let $\mathcal{F} = \{f_{\mathsf{k}} : \{0,1\}^n \to \{0,1\}^*\}_{\mathsf{k}\in\{0,1\}^n, n\in\mathbb{N}}$ be a family of robust unobfuscatable functions. For simplicity, let us also assume that the family is $\frac{2}{3}$-robust with respect to the uniform distribution ensemble $\mathcal{D} = \mathcal{U}$ (the proof easily generalizes to arbitrary $1-\epsilon$, and samplable distribution ensemble $\mathcal{D}$). We define a one-way function $\varphi = \{\varphi_n\}$ as follows: $\varphi_n : \{0,1\}^{n(m+1)} \to \{0,1\}^*$, and

$$(\mathsf{k}, x_1, \ldots, x_m) \xmapsto{\varphi_n} (f_{\mathsf{k}}(x_1), \ldots, f_{\mathsf{k}}(x_m), x_1, \ldots, x_m) \quad ,$$

where $m(n) = 2n$.

We next show that φ is one-way. For any $\mathsf{k} \in \{0,1\}^n$, we denote by z_{k} its corresponding unique unlearnable secret. First note that for any k, k' such that $\mathsf{z}_{\mathsf{k}} \neq \mathsf{z}_{\mathsf{k}'}$, the agreement between f_{k} and $f_{\mathsf{k}'}$ is at most $2/3$. Indeed, if that was not the case, we could construct a circuit $C_{\mathsf{k}\cap\mathsf{k}'}$ that given input x returns $f_{\mathsf{k}}(x)$, only if $f_{\mathsf{k}}(x) = f_{\mathsf{k}'}(x)$, and otherwise returns $\perp$. This circuit computes both functions with probability at least $2/3$ (without making errors); however, in this case the robust extraction guarantee, implies that the extractor outputs $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k}) \cap \mathcal{R}_{\mathcal{F}}(\mathsf{k}') = \{\mathsf{z}_{\mathsf{k}}\} \cap \{\mathsf{z}_{\mathsf{k}'}\}$, which means that $\mathsf{z}_{\mathsf{k}} = \mathsf{z}\mathsf{k}'$.

Thus we can deduce that for any $\mathsf{k} \in \{0,1\}^n$:

$$\Pr_{x_1,\ldots,x_m}\left[\exists \mathsf{k}' \;\middle|\; \begin{array}{c} \mathsf{z}_{\mathsf{k}} \neq \mathsf{z}_{\mathsf{k}'} \\ \forall i \in [m] : f_{\mathsf{k}}(x_i) = f_{\mathsf{k}'}(x_i) \end{array}\right] \leq |\left\{\mathsf{k}' \in \{0,1\}^n\right\}| \cdot \left(\frac{2}{3}\right)^m \leq \left(\frac{8}{9}\right)^n \quad .$$

It follows that any algorithm that inverts φ with noticeable probability, outputs k' such that $\mathsf{z}_{\mathsf{k}'} = \mathsf{z}_{\mathsf{k}}$; in particular, such an algorithm directly implies a learner that breaks the black-box unlearnability of $\mathcal{F}$ with respect to its unlearnable relation $\mathcal{R}_{\mathcal{F}}$. This learner would simply query its oracle f_{k} on m random points, run the inverter to obtain k' and compute $\mathsf{z}_{\mathsf{k}'} = \mathsf{z}_{\mathsf{k}}$. □

Remark 3.2. We note that the above lemma also holds if we consider a weaker form of robust unobfuscatable Turing machine families, whereas for non-robust unobfusactable Turing machine families can be constructed without any computational assumptions (see [BGI$^+$01]). In addition, the above also holds given a weaker inefficient extraction guarantee.

3.5 More Efficient Extraction from Fully Homomorphic Encryption

In this section, we discuss the running time of the extractor and its importance. We analyze the running of the extractor corresponding to Construction 3.1, and also present an augmented construction with more efficient extraction based on fully homomorphic encryption.

The extractor $\widehat{\mathsf{E}}$ given by Construction 3.3 evaluates each of the functions $\widehat{\mathbb{R}}^{\triangle}_{\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{a,b,\mathsf{s}}, \widehat{\mathbb{R}}^{\triangle}_{\mathsf{sk},b,\mathsf{s}}$, implemented by the input circuit C, a $\mathrm{poly}(n)$ number of times (where poly is some fixed polynomial). However, some of these evaluations are performed by "homomorphically evaluating" the circuit C on certain encryptions. Since each homomorphic operation is performed by evaluating the function $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ implemented by C, we get that the total running of $\widehat{\mathsf{E}}$ is $\mathrm{poly}(n) \cdot |C|^2$.

As discussed in Section 1.4 and will be further discussed in Section 5.3.2, improving the ratio between the running time of $\widehat{\mathsf{E}}$ and $|C|$ has applications to reducing the round complexity of resettably-sound and simultaneously resettable ZK. By replacing the symmetric encryption in Construction 3.1 with fully homomorphic encryption we can improve the running time of $\widehat{\mathsf{E}}$ to $\mathrm{poly}(n) \cdot |C|$; namely, we can make the dependency on $|C|$ linear instead of quadratic.

The main idea is that now $\widehat{\mathsf{E}}$ is able to emulate a homomorphic operation without evaluating the function $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ implemented by C.

The augmented construction. We start by overviewing the basic properties needed by the encryption scheme, compared to those of the previous one. First, we require that the scheme is fully-homomorphic and *rerandomizable* in the following sense:

Definition 3.4 (Rerandomizable encryption)**.** *an encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is rerandomizable if there exist an efficient ciphertext rerandomization algorithm* ReRan*, such that, for every sequence of ciphers and secret keys,* $\{\mathsf{c}_n, \mathsf{sk}_n\}_{n\in\mathbb{N}}$*:*

$$\{\mathsf{ReRan}(\mathsf{c}_n)\}_{n\in\mathbb{N}} \approx_s \{\mathsf{Enc}_{\mathsf{sk}_n}(\mathsf{Dec}_{\mathsf{sk}_n}(\mathsf{c}_n)))\}_{n\in\mathbb{N}} \quad .$$

(statistical indistinguishability can be naturally relaxed to computational)

In our modified construction, the learner will not have access to a function that performs homomorphic ciphertext operations and therefore we no longer require that the encryption is CCA-1. Additionally, our previous requirement of random ciphertext generation can be replaced by the above rerandomization property (which is needed anyhow). Specifically, we assume WLOG that, when constructing publicly verifiable robust unobfuscatable functions (see Section 4), the public verification key contains encryptions of the constants in $\{0, 1\}$. Now, using encryptions of constants and the ciphertext rerandomization algorithm, it is possible to sample random ciphertexts encrypting random bits. (The notion of publicly verifiable robust unobfuscatable functions we get from this construction is slightly weaker then defined in Definition 4.1 since the distribution $\mathcal{D}$ is only samplable given the public verification key. However, this suffices for all the applications to ZK, where the public verification key is known before evaluating the function.)

Recall that for Construction 3.1 we required that the underlying encryption scheme is decomposable (according to Definition 3.2). We will make the same requirement for the augmented construction; however, we next show that any homomorphic encryption scheme can be also made decomposable. If the original scheme is rerandomizable, so is the resulting scheme.

Construction 3.4 (Making a homomorphic encryption decomposable). Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a fully homomorphic encryption scheme. We construct a decomposable fully homomorphic encryption scheme $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ as follows:

- Gen' is identical to Gen.
- Enc' on a secret key sk and plaintext $b \in \{0, 1\}$, samples a random bit r from $\{0, 1\}$ and outputs $\mathsf{Enc}_{\mathsf{sk}}(r)||b \oplus r$.

- Dec' on a secret key sk and ciphertext $c||r$, outputs $r \oplus \mathsf{Dec}_{\mathsf{sk}}(c)$.
- Eval' on a pair of ciphertexts $c_1||r_1$ and $c_2||r_2$ samples a random bit r from $\{0,1\}$ and outputs $c||r$ where the ciphertext c is computed homomorphically (using Eval) from (c_1, c_2, r_1, r_2, r) according the homomorphic operation.
- If the original scheme has a rerandomization algorithm ReRan, the new scheme has a rerandomization algorithm ReRan' that on input ciphertext $c_1||r_1$, samples a random bit r from $\{0,1\}$, homomorphically computes a cipher c such that $\mathsf{Dec}(c_1||r_1) = \mathsf{Dec}(c||r)$ and outputs $\mathsf{ReRan}(c)||r$.

The algorithm pub on input ciphertext $c||r$ outputs c. (As remarked in Section 3.1, the algorithm $\mathsf{PubSamp}$ can be implemented by sampling a random cipher of a random plaintext and applying pub.)

Theorem 3.2. *Assuming rerandomizable homomorphic encryption, there exist a family of robust unobfuscatable functions, where the running time of the extractor* $\mathsf{E}(C, 1^n)$ *is* $\mathrm{poly}(n) \cdot |C|$.

Construction 3.5 (Robust unobfuscatable functions with efficient extraction). The construction is a variant of Construction 3.1 with the following modifications:

- The encryption scheme is a rerandomizable decomposable fully homomorphic bit encryption.
- The function f_{k} will no longer evaluate the function $\widehat{\mathbb{H}}_{\mathsf{sk}}$
- When making the function $\mathbb{A}_{\mathsf{sk},a}$ deterministic, a standard PRF can be used (instead of one that is invoker randomizable). In addition, the n-fold $\widehat{\mathbb{A}}_{\mathsf{sk},a}$ can be replaced with the 1-fold version $\mathbb{A}_{\mathsf{sk},a}$.

Proof sketch of Theorem 3.2. The proof of black-box unlearnability is similar to the proof of Lemma 3.1. In the proof of non-black-box learnability, the main modification we introduce to the extractor given by Construction 3.3 is in the emulation of ciphertext operations. Specifically, ciphertexts are no longer represented by n-fold ciphers; instead, values of intermediate wires in the homomorphic evaluation performed by $\widehat{\mathsf{E}}$ are represented by a single ciphertext, and homomorphic operations are performed using Eval (instead of using the function $\widehat{\mathbb{H}}^{\Delta}_{\mathsf{sk},\mathsf{s}'}$). Also, the circuit C_b is given one encryption of b instead of n encryptions. C_b will now use the ciphertext rerandomization algorithm ReRan to generate n random encryptions of b. □

4 Publicly Verifiable Robust Unobfuscatable Functions

In this section, we define publicly-verifiable robust unobfuscatable functions, and show, in Section 4.1, how to construct them from robust unobfuscatable functions (using a general compiler). We also show that such functions imply error-robust unobfuscatable functions. Throughout, we shall simply call them verifiable (rather than publicly-verifiable).

At high-level, a verifiable robust unobfuscatable family $\mathcal{F}$ is associated with a key generation algorithm $\mathsf{Gen}_{\mathcal{F}}$ that samples a secret key k and a (public) verification key vk. The verification key has two purposes:

- **Public verification of the unlearnable relation.** vk allows verifying a witness for a unlearnable relation $\mathcal{R}_{\mathcal{F}}$; namely, given z, it can be efficiently checked whether $\mathcal{R}_{\mathcal{F}}(\mathsf{vk}, \mathsf{z}) = 1$ (whereas in Definition 2.2, the secret key k is required for verification).
- **Public verification of an image property.** vk allows to publicly verify a given property of a candidate image a for $f_{\mathsf{k}}(q)$, for any given input q. Specifically, the family is associated with an efficient relation $\mathsf{Ver}_{\mathcal{F}}$, where $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1$ only if (q, a) satisfy a given property with respect to f_{k}. For example, f_{k} may consist of two functions $f_{\mathsf{k}_1}, f_{\mathsf{k}_2}$, and $\mathsf{Ver}_{\mathcal{F}}$ may try to verify that a pair $(q_1, a_1), (q_2, a_2)$ is such that either $a_1 = f_{\mathsf{k}_1}(q_1)$ or $a_2 = f_{\mathsf{k}_2}(q_2)$. We will require that $\mathsf{Ver}_{\mathcal{F}}$ has a completeness property, implying that it is always that case that if $a = f_{\mathsf{k}}(q)$, then $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a)$.

The black-box unlearnability property is defined just as for robust unobfuscatable functions, except that it should hold also against learners that are given the verification key vk. The non-black-box learnability property is strengthened: in the definition of robustness, the extractor was required to work for any C that agrees with the function on a given distribution $\mathcal{D} = \mathcal{D}(1^n)$; now we require that extraction works, even if the circuit C may never produce $a = f_{\mathsf{k}}(q)$, but does produce a's such that $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1$ with high-probability over inputs drawn from $\mathcal{D}$. In particular, the circuit C is allowed to output undetectable errors $a \notin \{f_{\mathsf{k}}(q), \perp\}$.

We now proceed to formally define the notion.

Definition 4.1 (A verifiable robust unobfuscatable function)**.** *A family of functions $\mathcal{F}$, with efficient key generation algorithm $\mathsf{Gen}_{\mathcal{F}}$, and image verification relation $\mathsf{Ver}_{\mathcal{F}}$, is a verifiable robust unobfuscatable family with respect to efficient relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$ if it is:*

1. **Black-box unlearnable:** *For any poly-size learner $\mathsf{L} = \{\mathsf{L}_n\}_{n\in\mathbb{N}}$:*

$$\Pr_{(\mathsf{k},\mathsf{vk})\leftarrow\mathsf{Gen}_{\mathcal{F}}(1^n)}[(\mathsf{vk},\mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{L}_n^{f_{\mathsf{k}}}(\mathsf{vk})] \leq \mathrm{negl}(n) \ .$$

   Non-Black-box learnable: *There exists an efficient extractor E such that for any noticeable function $\epsilon(n) = n^{-O(1)}$, all large enough $n \in \mathbb{N}$, any secret key and verification key $(\mathsf{k}, \mathsf{vk}) \in \mathrm{supp}(\mathsf{Gen}_{\mathcal{F}}(1^n))$, and every circuit C such that*

$$\Pr_{q\leftarrow\mathcal{D}(1^n)}[\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1 : C(q) = y] \geq \epsilon(n) \ ,$$

   E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{vk})$ from C:

$$\Pr_{\mathsf{E}}\left[(\mathsf{vk},\mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, \mathsf{vk}, 1^n, 1^{1/\epsilon(n)})\right] \geq 1 - \mathrm{negl}(n) \cdot \mathrm{poly}(|C|) \ .$$

2. **Verification completeness:** *For any $(\mathsf{k}, \mathsf{vk}) \in \mathrm{supp}(\mathsf{Gen}_{\mathcal{F}}(1^n))$, and for any q in the domain of f_{k}, $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, f_{\mathsf{k}}(q)) = 1$.*

It is not hard to see that verifiable robust unobfuscatable functions are, in particular, stronger than error-robust unobfuscatable functions.

Lemma 4.1. *Any verifiable robust unobfuscatable function family $\mathcal{G}$ can be transformed into an error-robust unobfuscatable function family $\mathcal{F}$ (according to Definition 2.3).*

Proof sketch. The transformation from $\mathcal{G}$ to $\mathcal{F}$: a key k' for a function $f_{\mathsf{k}'} \in \mathcal{F}$ consists of a key k and a corresponding verification key vk for a function g_{k}. The input sampler $\mathcal{D}$ is the same as for $\mathcal{F}$, and a function $f_{\mathsf{k}'}$ is defined as follows: given an input $q \leftarrow \mathcal{D}(1^n)$, $f_{\mathsf{k}'}(q)$ returns $(g_{\mathsf{k}}(q), \mathsf{vk})$, the unlearnable relation $\mathcal{R}_{\mathcal{F}}$ will be $\{((\mathsf{k}, \mathsf{vk}), \mathsf{z}) : \mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk})\}$. First, black-box unlearnability of the constructed $\mathcal{F}$ follows directly from that of $\mathcal{G}$. Second, for black-box non-learnability, note that if a circuit C $(\frac{1}{2} + \epsilon, \mathcal{D})$-approximates $f_{\mathsf{k}'}$, even with errors (where $\mathcal{D} = \mathcal{D}(1^n)$). Then, an extractor $\mathsf{E}_{\mathcal{F}}$ can first sample C (about n/ϵ) to identify, with overwhelming probability, the single verification key vk, such that $\mathsf{k}' = (\mathsf{vk}, \mathsf{k})$; then, it can construct from C a new circuit C' that $(\frac{1}{2} + \epsilon, \mathcal{D})$-approximates f_{k} in the sense of Definition 4.1; indeed, the completeness of $\mathsf{Ver}_{\mathcal{F}}$ says that whenever C agrees with the function $\mathsf{Ver}_{\mathcal{F}}$ will accept. Now, the extractor can run the underlying extractor $\mathsf{E}_{\mathcal{G}}$ and obtain $\mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk})$. □

4.1 Constructing Verifiable Robust Families

We now show how to construct verifiable robust families from robust families with a hardcore secret (Definition 2.5), which in turn can be constructed from any robust family with a unique-witness relation (Lemma 2.3), such as the family constructed in Section 3.

Theorem 4.1. *Assuming, trapdoor permutations exist, family $\mathcal{G}$ that is a robust unobfuscatable family with respect to a hardcore family $\{\mathcal{HC}_n\}$, can be compiled into a verifiable robust family $\mathcal{F}$ (according to Definition 4.1).*

Proof sketch. We prove the theorem in two steps: we first show how to get a verifiable family that only satisfies public-verifiability of the unlearnable relation, but will only be robust against detectable errors (like the family $\mathcal{G}$ we started from). Then, we show how to transform the resulting family into one that also satisfies public-verifiability of some prescribed image property, and will already be robust to errors, given that the property is frequently satisfied.

Step 1 - a family with a publicly-verifiable unlearnable relation. Given a family $\mathcal{G}$ with a hardcore family $\{\mathcal{HC}_n\}$, we augment each function $g_{\mathsf{k}} \in \mathcal{G}$ and its corresponding (secret) key k, with public key $\mathsf{vk} = (h, y)$, where $h \leftarrow \mathcal{HC}_n$ is randomly chosen hardcore function, and $y = \varphi(h(\mathsf{k}))$ for a one-way function φ. We then define the publicly verifiable unlearnable relation to be

$$\mathcal{R}_{\mathcal{G}} = \{(\mathsf{vk}, \mathsf{z}) = ((h, y), x) \mid y = \varphi(x)\} \quad .$$

Claim 4.1. *A randomly chosen $g_{\mathsf{k}} \in \mathcal{G}$ is black-box unlearnable with respect to the relation $\mathcal{R}_{\mathcal{G}}$, even given a corresponding vk. Also, for any $h \in \mathcal{HC}_n$, and k, $h(\mathsf{k})$ is non-black-box learnable from any circuit C that $(\mathcal{D}, \epsilon)$-approximates f_{k} is the input distribution corresponding to $\mathcal{G}$).*

Proof sketch. First, to see that g_{k} is unlearnable with respect to $\mathcal{R}_{\mathcal{G}}$, recall that $h(\mathsf{k})$ is pseudo random, even given black-box access to g_{k}, and h. Thus, any learner that manages to satisfy $\mathcal{R}_{\mathcal{G}}$, would also manage to do so had we given it $\varphi(u)$, contradicting the one-wayness of φ.

The fact that $\mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{k})$ can be extracted, given a circuit that $(\mathcal{D}, \epsilon)$-approximates g_{k}, and $\mathsf{vk} = (h, y)$, directly follows from the extraction guarantee of $\mathcal{G}$. Indeed, $\mathsf{E}_{\mathcal{G}}$, given h and C, is guaranteed to extract $h(\mathsf{k})$, and thus produces the required pre-image in $\varphi^{-1}(y)$. □

Step 2 - public-verification of an image property. The family $\mathcal{G}$, after being augmented as above, still does not have a publicly-verifiable image property. We now construct from $\mathcal{G}$ a new family $\mathcal{F}$ that will already satisfy all the requirements of a verifiable robust family as given by Definition 4.1.

The basic idea is to embed, in the function's answers, rZAPs (see Section 5.1) attesting that the answer is consistent with the verification key; however, to securely use rZAPs, we first need to establish a proper statement with at least two valid witnesses. We use a similar idea to that used in [FS90, COSV12], of using a WI proof (or rZAPs in our case) in order to get a witness-hiding proof for statements with two independent witnesses. Details follow.

Each function f_{k} in the family $\mathcal{F}$ consists of two independent functions g_{k_0} and g_{k_1} sampled independently from $\mathcal{G}$, and randomness s for an rZAP; the secret key k is set to be $(\mathsf{k}_0, \mathsf{k}_1, \mathsf{s})$. The corresponding verification key vk, will consist of the two verification keys $(\mathsf{vk}_0, \mathsf{vk}_1)$ sampled with the above to functions (as defined in Step 1), as well as two commitments $\mathsf{C}_0 = \mathsf{Com}(\mathsf{k}_0)$, $\mathsf{C}_1 = \mathsf{Com}(\mathsf{k}_1)$, and a third commitment $\mathsf{C} = \mathsf{Com}(b)$, to a random bit $b \leftarrow \{0,1\}^n$ (the commitments are non-interactive and perfectly binding as defined in Section 5.1); overall, $\mathsf{vk} = (\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$. The input sampler for $\mathcal{F}$ is given by the product $\widehat{\mathcal{D}} = \mathcal{D} \times \mathcal{U}$; namely it consists of an input for the underlying $g_{\mathsf{k}_0}, g_{\mathsf{k}_1}$, and a random string r, which will be a uniformly random first message for a rZAP.

The function f_{k}, given an input (q, r), computes $a_0 = g_{\mathsf{k}_0}(q)$ and $a_1 = g_{\mathsf{k}_1}(q)$, and then treating r as the first message of an rZAP, and using the rZAP randomness s, computes an rZAP proof for the statement:

$$\left\{\{\mathsf{C} = \mathsf{Com}(0)\} \vee \left\{ \begin{array}{c} \mathsf{C}_0 = \mathsf{Com}(\mathsf{k}_0) \\ a_0 = g_{\mathsf{k}_0}(q) \end{array} \right\}\right\} \bigwedge \left\{\{\mathsf{C} = \mathsf{Com}(1)\} \vee \left\{ \begin{array}{c} \mathsf{C}_1 = \mathsf{Com}(\mathsf{k}_1) \\ a_1 = g_{\mathsf{k}_1}(q) \end{array} \right\}\right\} \ ;$$

This statement has two witnesses: one witness consists of the bit b and randomness for the commitment C, as well as the key k_{1-b}, and randomness for the commitment C_{1-b}. the second witness consists of the two keys $\mathsf{k}_0, \mathsf{k}_1$ and the randomness corresponding to both commitments $\mathsf{C}_0, \mathsf{C}_1$. The function f_{k} correctly computes $a_b = g_{\mathsf{k}_b}(q)$, for both $b \in \{0, 1\}$, and gives the proof using the second witness. Finally, the unlearnable relation for the new family $\mathcal{F}$ will be:

$$\mathcal{R}_{\mathcal{F}} = \{((\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C}), \mathsf{z}) \mid \mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk}_0) \vee \mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk}_1)\} \ ;$$

namely, z is a witness if it is a witness for either one of the underlying verification keys vk_0 or vk_1.

Claim 4.2. *The function family $\mathcal{F}$ is verifiable and robust according to Definition 4.1.*

Proof sketch. First, we show that a random $f_{\mathsf{k}} \in \mathcal{F}$ is black-box unlearnable, even given the verification key vk. Specifically, we show that any learner L that satisfies $\mathcal{R}_{\mathcal{F}}(\mathsf{vk})$ with noticeable probability $\epsilon = \epsilon(n)$ can be transformed into a learner L' for the family $\mathcal{G}$. Indeed, given oracle access to a random g_{k}, and its verification key vk', L' will treat its oracle g_{k} as g_{k_b} and the verification key vk' as vk_b. In addition, L' samples a random bit b, and then samples $g_{\mathsf{k}_{1-b}} \in \mathcal{G}$ on its own, together with a verification key vk_{1-b}. L' then feeds the learner L with a verification key vk consisting of commitments $\mathsf{C} = \mathsf{Com}(b), \mathsf{C}_{1-b} = \mathsf{Com}(\mathsf{k}_{1-b})$ and $\mathsf{C}_b = \mathsf{Com}(0^{|\mathsf{k}_b|})$, and the verification keys $\mathsf{vk}_b, \mathsf{vk}_{1-b}$. Now, L' emulates L, answering any query (q, r) using its own sampled function for answer a_{1-b}, and the oracle for answer a_b. The proof is then given using what we referred above a "the first witness"; namely, for one part of the "and" statement b and the randomness for $\mathsf{C} = \mathsf{Com}(b)$ is used, and for the second part the function $g_{\mathsf{k}_{1-b}}$ and the randomness for $\mathsf{C}_{1-b} = \mathsf{Com}(\mathsf{k}_{1-b})$ is used.

We now claim that with probability at least $\epsilon/2$, the emulated L (and thus also L') outputs $\mathsf{z}_b \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk}_b)$ for the oracle g_{k_b}. Indeed, using the fact that the commitment Com is hiding, and that the rZAP is (resettably) witness-indistinguishable, we can use a standard hybrid argument to move to an experiment in which L has the exact same view as in a true interaction with f_{k}, where it outputs z that satisfies the unlearnable relation corresponding to one of the two functions. Then, using again the hiding of the commitment, we can move to an experiment where C is a commitment to a bit b' that is chosen independently of b, and L still succeeds; thus, it will output the right bit at least in $\frac{1}{2}$ of its successes.

We now show that $\mathcal{F}$ is non-black-box learnable according to Definition 4.1. Specifically, we define the image relation $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, (q, r), (a_0, a_1, \pi))$ to be a relation that simply verifies the rZAP proof π with respect to a first random message r, and the statement given by $(\mathsf{vk}, q, a_0, a_1)$. (Clearly, $\mathsf{Ver}_{\mathcal{F}}$ has the required completeness: $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, (q, r), f_{\mathsf{k}}(q, r)) = 1$ for any triple). Next, for $\epsilon = n^{-O(1)}$, let C be any circuit that $(\widehat{\mathcal{D}}, \epsilon)$-approximates f_{k} in the weak sense given by Definition 4.1. The extractor $\mathsf{E}_{\mathcal{F}}$ will operate as follows: it will first transform C into two circuits C_0 and C_1 such that one of them will $(\mathcal{D}, \epsilon - n^{-\omega(1)})$-approximate the underlying function g_{k_0} or g_{k_1}. The circuit C_b given a sample q from $\mathcal{D}(1^n)$, samples randomness r for an rZAP and feeds (q, r) to C, when it gets a result (a_0, a_1, π), it verifies the proof π, if it does not verify, C_b outputs $\perp$, otherwise it outputs a_b. Let b be the plaintext bit in the commitment C given in the verification key vk. We claim that with overwhelming probability over the choice of randomness r for the rZAP, C_b $(\mathcal{D}, \epsilon - n^{-\omega(1)})$-approximates g_{k_b}. Indeed, by taking the randomness for the rZAP to be of sufficiently large length $\mathrm{poly}(n)$, it can be guaranteed that with overwhelming probability all false statements (of some fixed bounded length $\ell(n) \ll \mathrm{poly}(n)$) are rejected (see [DN07]). Therefore, C_b only makes detectable errors (i.e., outputs $\perp$). Also, we are

guaranteed that C satisfies $\mathsf{Ver}_{\mathcal{F}}$ (and, in particular, satisfies the rZAP statement) with probability at least ϵ, and thus $(\widehat{\mathcal{D}}, \epsilon - n^{-\omega(1)})$-approximation follows as required. Hence, it suffices that the extractor $\mathsf{E}_{\mathcal{F}}$ chooses randomness for both C_0, C_1 and feeds each one of them to the extractor $\mathsf{E}_{\mathcal{G}}$ of the family $\mathcal{G}$, and it will manage to produce $\mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}_b)$. □

This concludes the proof of Theorem 4.1 □

5 From Verifiable Robust Unobfuscatable Functions to Resettable Protocols

5.1 Definitions

We briefly recall the definitions of resettable ZK [CGGM00], resettable soundness [BGGL01], and several other basic definitions and tools used in this section. Most definitions are taken almost verbatim from [CGGM00, BGGL01, DGS09]. In Subsection 5.1.1, we define instance-dependent resettable witness-indistinguishability.

Resettable ZK. we start by recalling resettable ZK.

Definition 5.1 (Resettable zero-knowledge [CGGM00])**.** *An interactive proof system (P, V) for a language $\mathcal{L}$ is said to be resettable zero-knowledge if for every probabilistic polynomial-time adversary V^* there exists a probabilistic polynomial time simulator $\mathcal{S}$ so that the distribution ensembles $\mathcal{D}_1$ and $\mathcal{D}_2$ described below are computationally indistinguishable: Let each distribution be indexed by a sequence of distinct common inputs $\vec{x} = x_1, \ldots, x_{\mathrm{poly}(n)} \in \mathcal{L} \cap \{0,1\}^n$ and a corresponding sequence of prover's auxiliary-inputs $\vec{y} = y_1, \ldots, y_{\mathrm{poly}(n)}$.*

Distribution $\mathcal{D}_1$ is defined by the following random process which depends on P and V^*

- *Randomly select and fix $t = \mathrm{poly}(n)$ random-tapes $\omega_1, \ldots, \omega_t$ for P, resulting in deterministic strategies $\mathsf{P}^{(i,j)} = \mathsf{P}_{x_i, y_i, \omega_j}$ defined by $\mathsf{P}_{x_i, y_i, \omega_j}(\alpha) = \mathsf{P}(x_i, y_i, \omega_j, \alpha)$ for $i, j \in \{1, \ldots, t\}$. Each $\mathsf{P}^{(i,j)}$ is called an incarnation of P.*
- *Machine V^* is allowed to run polynomially-many sessions with the $\mathsf{P}^{(i,j)}$'s. Throughout these sessions, V^* is required to complete its current interaction with the current copy of $\mathsf{P}^{(i,j)}$ before starting a new interaction with any $\mathsf{P}^{(i', j')}$, regardless if $(i, j) = (i', j')$ or not. Thus, the activity of V^* proceeds in rounds. In each round it selects one of the $\mathsf{P}^{(i,j)}$'s and conducts a complete interaction with it.*
- *Once V^* decides it is done interacting with the $\mathsf{P}^{(i,j)}$'s it (i.e, V^*) produces an output based on its view of these interactions. This output is denoted by $(\mathsf{P}(\vec{y}), \mathsf{V}^*)(\vec{x})$ and is the output of the distribution.*

Distribution $\mathcal{D}_2$: The output of $\mathcal{S}(\vec{x})$.

Resettably-sound ZK. We briefly recall the definitions of resettable-soundness presented in [BGGL01]. In the setting of resettable-soundness, the prover P^* has the power to reset the verifier V. Specifically, the random tape of V is chosen at random and fixed once and for all and, from that point on, the prover can interact multiple times with the residual deterministic verifier $\mathsf{V}_{\mathsf{r}}(x)$ induced by r and the common input x. Each such interaction is called a session.

Note that the adversary may repeat in a current session the same messages sent in a prior session, resulting in an identical prefix of an interaction (since the verifier's randomness is fixed). Furthermore, by deviating in the next message, the adversary may obtain two different continuations of the same prefix of an interaction.

A generalization of the above model, also considered in [BGGL01], is to allow the prover to interact with multiple "incarnations" of the verifier. Here, $t = \mathrm{poly}(n)$ random tapes $\mathsf{r}_1, \ldots, \mathsf{r}_t$ are sampled, and the prover can adaptively choose at any point an input x and index $i \in [t]$ and interact with $\mathsf{V}_{\mathsf{r}_i}(x)$.

Definition 5.2 (Resettably sound argument[BGGL01]). *A resetting attack of a malicious prover* P^* *on a resettable verifier* V *is defined by the following random process, indexed by a security parameter* n*:*

1. *Uniformly select pick* $t = \mathrm{poly}(n)$ *random-tapes, denoted* $\mathsf{r}_1, \ldots, \mathsf{r}_t$ *for* V*, resulting in deterministic strategies* $\mathsf{V}_{\mathsf{r}_1}, \ldots, \mathsf{V}_{\mathsf{r}_t}$*. For a given* $x \in \{0,1\}^n$*,* $\mathsf{V}_{\mathsf{r}_i}(x)$ *is called an incarnation.*

2. *A prover* P^* *of size* $\mathrm{poly}(n)$ *can initiate* $\mathrm{poly}(n)$ *interactions with different incarnations. In each such interaction,* P^* *chooses* $x \in \{0,1\}^n$ *and* $i \in [t]$*, and conducts a complete session with the incarnation* $\mathsf{V}_{\mathsf{r}_i}(x)$*.*

An argument system $\langle \mathsf{P} \leftrightarrows \mathsf{V} \rangle$ *is a resettably sound argument for* $\mathcal{L}$*, for any resetting poly-size* P^**, the probability that in some session during a resetting attack,* P^* *convinces some incarnation* $\mathsf{V}_{\mathsf{r}_i}(x)$ *of accepting while* $x \notin \mathcal{L}$ *is negligible in* n*.*

For simplicity, we concentrate on the simple case of one incarnation $\mathsf{V}_{\mathsf{r}}(x)$; however, all of our results directly extend to the model of multiple incarnations.

Resettably-sound witness-indistinguishable arguments of knowledge. An rsWIAOK, is WI in the usual since, and is also an argument of knowledge against resetting provers; namely we can efficiently extract a witness from any resetting prover that convinces the verifier of accepting with noticeable probability. Such proof systems can be constructed from classical public-coin proof systems such as (the n-fold version of) Blum's Hamiltonicity WI protocol [Blu86] by applying to them the [BGGL01] transformation (the resulting constructions are based solely on one-functions).

A useful property of these classical protocols is that the knowledge extraction uses the prover as a black-box (except for being given its size).

Resettable ZAPs. ZAPs are 2-message public-coin witness-indistinguishable proofs introduced by Dwork and Naor [DN07]. They further have the special property that the first message (sent by the prover) can be reused for multiple proofs. As noted in [BGGL01], any ZAP already has the property of resettable soundness. Furthermore, resettable witness-indistinguishability property can be obtained by applying the transformation in [CGGM00]. We refer to the resulting system as an rZAP system having the property of resettable-soundness as well as resettable-witness indistinguishability.

Commitments. In this work, we shall use two types of perfectly (or statistically) binding commitments. One type is non-interactive commitments, which can be constructed based on any one-way permutation (let alone a trapdoor permutation) [Blu81]. The second are interactive (2-message) commitments that can be constructed from one-way functions [Nao91].

5.1.1 Instance-dependent resettable witness-indistinguishable arguments.

An instance-dependent resettable-WI argument (rWI^y), for an NP language $\mathcal{L}$, is defined with respect to a candidate instance y for another (possibly different) NP language $\mathcal{L}'$. In an instance-dependent rWI^y the soundness and rWI properties are decoupled according to y: if $y \in \mathcal{L}'$ the system is a sound and if $y \notin \mathcal{L}'$ the system is rWI.

Definition 5.3 (rWI^y). *An argument system* $\langle \mathsf{P} \leftrightarrows \mathsf{V} \rangle$ *for* $\mathcal{L}$ *is rWI with respect to* $\mathcal{L}'$ *if satisfies:*

1. **Instance-dependent soundness:** *for any poly-size prover* P^**, and all large enough* $x \in \{0,1\}^n \backslash \mathcal{L}$ *and* $y \in \mathcal{L}'$*:*

$$\Pr_{\mathsf{V}}[\langle \mathsf{P}^* \leftrightarrows \mathsf{V} \rangle(x, y) = 1] \leq \mathrm{negl}(n) \ .$$

2. **Instance-dependent rWI:** *for any poly-size resetting verifier* V^**, all large enough* $x \in \mathcal{L}$*,* $w_x^{(1)}, w_x^{(2)} \in \mathcal{R}_{\mathcal{L}}(x)$*, and* $y \notin \mathcal{L}'$*:*

$$\Pr\left[\langle \mathsf{P}(w_x^{(1)}) \leftrightarrows \mathsf{V}^* \rangle(x,y) = 1\right] - \Pr\left[\langle \mathsf{P}(w_x^{(2)}) \leftrightarrows \mathsf{V}^* \rangle(x,y) = 1\right] \leq \mathrm{negl}(n) \ .$$

Constructing instance-dependent rWIs from one-way functions. Instance-dependent rWI can be constructed based on the [CGGM00] transformation, and instance-dependent commitment schemes (also known as equivocable commitments), which in turn can be constructed from one-way functions [FS89, Dam89].

We briefly recall what are the properties of such commitments (a more formal definition can be found in [FS89]). An instance-dependent commitment scheme is also parameterized by an instance y for an NP language $\mathcal{L}'$ and has two properties:

1. **Instance-dependent equivocation** There is an efficient equivocation algorithm Eq that, given $(y, w) \in \mathcal{R}_{\mathcal{L}'}$ and some length ℓ, can produce an equivocal commitment $\tilde{\mathsf{C}}$, which it can open to any $x \in \{0,1\}^\ell$ (the equivocation algorithm uses w). Equivocal commitments and their openings are computationally indistinguishable from honestly generated commitments and their openings.

2. **Instance-dependent binding** For $y \notin \mathcal{L}$, the commitment is statistically binding.

Given such commitments, it is possible to transform any public-coin WI system (e.g. Blum) to an instance-dependent rWI, using the [CGGM00] transformation. Specifically, given an instance y as a candidate for $\mathcal{L}$, the verifier first commits to its random message using the y-dependent commitment. Then, the parties run the WI protocol, where the verifier opens the commitment as its message. The randomness used by the prover is derived by applying a pseudo random function to the verifier's commitment. If $y \in \mathcal{L}'$, it follows by the equivocation guarantee, and the soundness for the underlying WI, that the protocol is sound. If $\notin \mathcal{L}$, the commitment is binding, and rWI follows as in [CGGM00].

5.2 The Base Protocol

In this section, we present an $O(m)$-round resettably sound ZK protocol where m is a parameter that will be chosen according to the desired notion of ZK (stand-alone or concurrent).

In what follows, let $\mathcal{F}$ be a verifiable robust unobfuscatable function family, with respect to the relation $\mathcal{R}_{\mathcal{F}}$. Let $\mathsf{Gen}_{\mathcal{F}}, \mathsf{Ver}_{\mathcal{F}}$ be the key generation algorithm and the image verification relation of $\mathcal{F}$, and let $\mathcal{D}$ be the efficient sampler guaranteed by the non-black-box learnability property of $\mathcal{F}$. The protocol is described in Figure 1.

5.3 The Resettable Security of the Protocol

In this section, we show analyze the resettable security of Protocol 1, according to the setting of the parameter m.

5.3.1 Resettable Soundness

We show that Protocol 1 is resettably sound, for any setting of m (constant or $\mathrm{poly}(n)$).

Lemma 5.1. *For any* m*, Protocol 1 is a resettably-sound ZK argument of knowledge.*

Proof sketch. Given a prover P^* that, for a set $\mathcal{X}$ of inputs, convinces V to accept each $x \in \mathcal{X} \cap \{0,1\}^n$ with probability $\epsilon(n)$. We can easily construct PPT extractor E that (for all but finitely many $x \in \mathcal{X}$) extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$. First we consider a new prover P_1^* against the rsWIAOK

Protocol 1

Common Input: $x \in \mathcal{L} \cap \{0,1\}^n$.

Auxiliary Input to P**:** $w \in \mathcal{R}_{\mathcal{L}}(x)$.

1. V samples keys $(\mathsf{k}, \mathsf{vk}) \leftarrow \mathsf{Gen}_{\mathcal{F}}(1^n)$ for $\mathcal{F}$ and sends vk to P.
2. Repeat the following function evaluation slot m times:
   (a) P samples $q \leftarrow \mathcal{D}(1^n)$ and sends q to V.
   (b) V evaluates the function $a = f_{\mathsf{k}}(q)$ and sends a to P.
   (c) P verifies that indeed $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1$.
3. P proves the following statement to V using an rsWIAOK:
   "$x \in \mathcal{L}$ or $\{\exists \mathsf{z} : (\mathsf{vk}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}}\}$".

Figure 1: A resettably-sound (concurrent) ZK protocol

protocol concluding the proof. The new prover P_1^* is given oracle access to a random $f_{\mathsf{k}} \in \mathcal{F}$, as well as a corresponding public key vk, and tries to convince the rsWIAOK verifier of the WI statement corresponding to x and vk. This prover perfectly emulates P^*, forwarding to the oracle f_{k} any query that P^* makes in the function evaluation slot, and forwarding any message in the proof stage to the external resettable verifier. Since the view of the emulated P^* is the same as in a real execution it convinces the rsWIAOK verifier with the same probability ϵ. We can now apply the black-box extractor E_{wi} for the rsWIAOK to extract a witness from P_1^*. To do so, E_{wi} can sample f_{k} and vk on its own, and use them to answer any oracle call that P_1^* makes. By the POK guarantee, we know that E_{wi} outputs a witness for the WI statement corresponding to x and vk (in expected polynomial time). It is left to see that, except with negligible probability, this witness will be $w \in \mathcal{R}_{\mathcal{L}}(x)$, rather than $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{vk})$; otherwise, we can use $\mathsf{E}_{\mathsf{wi}'}$ and P_1^* to break the unlearnability of $\mathcal{F}$. Indeed, since E_{wi} is a black-box extractor (see Section 5.1), it can answer all of P_1^*'s calls to the function, using an external oracle to f_{k}. □

5.3.2 Stand-Alone (Non-Resettable) ZK

In this section, we show that Protocol 1 for $m = O(1)$ is (stand-alone) ZK (in the next section, we'll show that for $m = n^{\Omega(1)}$, it is concurrent ZK).

Let $\mathcal{F}$ be a verifiable robust unobfuscatable function family, as used Protocol 1, and let d be a constant such that the running time of the non-black-box extractor of $\mathcal{F}$, given security parameter n, and approximation parameter ϵ, is $\mathrm{poly}(n) \cdot \left(\frac{|C|}{\epsilon}\right)^d$.

Lemma 5.2. *Protocol 1 with $m = d$ is ZK.*

We describe the main ideas behind the proof of Lemma 5.2. More details can be found in [BP12].

Proof sketch. Following the simulation technique of Goldreich and Kahan [GK96], the simulator starts by running the cheating verifier V^* once, simulating its view until the proof step. If the V^* does not abort and evaluates the function correctly in Step 3b, the simulator interacts with V^* repeatedly to estimate the probability ϵ_i that V^* evaluates the function correctly in each slot i. This estimation can be computed in expected polynomial time, and it can be shown that for some $i \in [d]$, it holds that $\epsilon_i \geq \epsilon^{1/d}$, where ϵ is the probability that V^* does not abort in any of the slots. The simulation runs the extractor E of the unobfuscatable function family on the circuit corresponding to the next messages function of V^* in a slot i as the above, using the approximation parameter ϵ_i. Correctness of the simulation follows from

the properties of the unobfuscatable function and the WI property of rsWIAOK. Since the extractor runs in time $\frac{\mathrm{poly}(n)}{\epsilon_i^d} \leq \frac{\mathrm{poly}(n)}{\epsilon}$ and since V^* aborts before the proof step with probability $1-\epsilon$ the expected running time of the simulation is $\mathrm{poly}(n)$. □

Round-efficient resettably-sound ZK. As explained in Section 3.5 the construction described in Section 3 either yields extractors with running time $\mathrm{poly}(n) \cdot |C|^2$, assuming trapdoor permutations, or extractors running in time $\mathrm{poly}(n) \cdot |C|$, assuming fully-homomorphic encryption. However, the size of the circuit $|C|$ in our protocol will be $|\mathsf{V}^*| \cdot \frac{\mathrm{poly}(n)}{\epsilon}$ (the $\frac{\mathrm{poly}(n)}{\epsilon}$ overhead is incurred by Lemma 2.1). Thus the actual running times are $\mathrm{poly}(n) \cdot \left(\frac{|\mathsf{V}^*|}{\epsilon}\right)^2$ and $\mathrm{poly}(n) \cdot \left(\frac{|\mathsf{V}^*|}{\epsilon}\right)$ (respectively).

Plugging this into Protocol 1, we would get by Lemma 5.2, an 8-message protocol in the first case and a 6-message one in the second case, if we use say a 3-message rsWIAOK (e.g. based on [Blu86]). However, we can, in fact, save a round and run the first two messages of the rsWIAOK in parallel to the last slot of the protocol (Step 3) similarly to the protocol of [FS89].

Corollary 5.1 (of Lemma 5.2)**.** *Assuming trapdoor permutations there exist a 6-message resettably-sound ZK protocol.*

Corollary 5.2 (of Lemma 5.2)**.** *Assuming rerandomizable fully homomorphic encryption and trapdoor permutations, there exist a 4-message resettably-sound ZK protocol.*

5.3.3 Concurrent and Resettable ZK

In this section, we show that for $m = n^{\Omega(1)}$ Protocol 1 is concurrent ZK. A simultaneously resettable ZK protocol can then be obtained from Protocol 1 by applying the general transformation of [DGS09] from resettably-sound concurrent ZK to simultaneously resettable ZK.

Lemma 5.3. *For every constant $\delta > 0$, Protocol 1 with $m = n^{\delta}$ is concurrent zero-knowledge.*

Proof of Lemma 5.3.

Overview of the simulation. The simulation uses the techniques of [RK99], and more specifically, the slightly augmented version of them applied in [DGS09, CLP10]. We start by describing the main ideas behind this technique as they where used in previous works. The execution of the protocol in every session consists of running many sequential slots giving the simulator many chances to "solve" the session. The simulation runs a *main thread*, and in the beginning of every slot, the simulation also starts "look-ahead" threads that are only supposed to simulate the interaction until the end of the slot. if the simulation in the look-ahead thread is successful, the simulator can then continue the simulation in the main thread and solve the session.

The main difficulty is that, in the concurrent setting, even the simulation of one slot may be non trivial. The idea is to have look ahead threads recursively use the same simulation strategy. If some slot contains too many other concurrent slots, the simulation of the corresponding look-ahead thread may "give up" and not reach the end of the slot. However, it can be shown that, in every session, some slots must be successfully simulated by look-ahead threads and therefore on the main thread, all sessions will be solved.

Our setting. In our simulation, we think of every slot as computing V^*'s function f_{k}. Instead of running a look-ahead thread to simulate the execution of a slot, we construct the circuit implementing such look-ahead thread and run the extractor of the unobfuscatable function family on this circuit. While we do not start any look ahead threads, the nature of our simulation is still recursive since the circuits we construct and extract from are also constructing smaller simulation circuits themselves and extracting from them.

We now describe several sub-routines used in our simulation procedure. We first describe them at high-level and then move on to describe them in more detail.

Let V^* be a concurrent verifier that opens at most n^c sessions, and assume WLOG that V^* is deterministic. We refer to every iteration of Step 3 where V^* evaluates the function f_{k} as a slot. In a single session, there are n^δ slots, and in the entire execution of V^* with the honest prover, there are at most $n^{c'} = n^{c+\delta}$ slots (less if V^* aborts or deviates from the protocol). The simulator $\mathcal{S}$, using the code of V^*, uses the following functions to create the simulated transcript:

The function MAIN simulates a given number of slots. MAIN takes the parameters (T, m) where T is the state of the concurrent simulation at the time the function is called, and m is the number of additional slots to simulate.

The function EMULATE transforms the code of V^* into a circuit that computes the function f_{k}. EMULATE is given a state T of a partial simulation of V^* right before the beginning of a slot. EMULATE generates a circuit C that, given input q, tries to evaluate the function $f_{\mathsf{k}}(q)$ by simulating the slot with the input q as the first message. C simulates V^* according to same logic as in the function MAIN until the slot terminates. EMULATE takes the parameters (T, m), and uses them within C in the same way that they are used in the MAIN function.

The function EXTRACT. This function tries to extracts the unlearnable secret z of the function f_{k}, in some session, by executing the extractor E of the unobfuscatable function family on the circuit C returned by a call to EMULATE. EXTRACT takes parameters (T, m) and passes them to EMULATE.

We note following recursive relations between the above functions:

- When $m > 1$ the function MAIN will execute the function EXTRACT on every slot it starts, passing it a smaller m.
- The function EXTRACT uses the circuit C generated the function EMULATE.
- The circuit C generated by the function EMULATE simulates messages according to the strategy of the function MAIN. In particular, C executes the function EXTRACT with a smaller value of m.

The initial call. The simulator $\mathcal{S}$ will execute the function MAIN with $m = n^{c'}$ and the empty simulation state.

Next, we describe the above functions in more detail.

The function MAIN. The function is given the state T of the simulation so far, and extends it as described above, updating the value of T as the simulation proceeds. Specifically, the simulation up to Step 3 (sending the rsWIAOK) in all sessions follows the strategy of the honest prover P (note that the witness w is not required for that). The state of the simulation is just the state maintained by P along the different sessions.

Whenever a slot starts, MAIN executes the function EXTRACT with parameters (T', m') where T' is the simulation state just before the beginning of the slot, and $m' = m/\Delta$ where $\Delta = n^{\delta'}$ for some $0 < \delta' < \delta$ a parameter of the simulation. If EXTRACT manages to output an unlearnable secret z such that $\mathcal{R}(\mathsf{vk}, \mathsf{z}) = 1$ for the vk sent in the session, then z is saved in T.

To simulate the rsWIAOK given in some session (Step 3), MAIN uses the unlearnable secret of the session saved in T. If no such secret is previously extracted MAIN aborts. In this case we say that the simulation "got stuck", and main will return $\perp$. When V^* terminates or starts the $m + 1$ slot, MAIN terminates and outputs the current value of T.

The function EMULATE. On parameters (T, m), the function creates a circuit C and outputs it. C simulates V^* by following almost the same logic as in the function MAIN with the same parameters (T, m). The only difference between the simulation of C and the function MAIN is that C get an external input q and sends it as the prover message in the first slot that V^* starts (when executed from T). If during the simulation the first slot closes, C obtains V^*'s response a and outputs it, otherwise it outputs $\perp$. As in the function MAIN, If V^* aborts or starts m additional slots, before the first slot terminates, C outputs $\perp$.

The function EXTRACT. On parameters (T, m), the function calls the function EMULATE with the same parameters and obtains a circuit C. EXTRACT then executes the extractor E of the unobfuscatable functions family on C with $\epsilon = \frac{1}{16}$ and outputs the same as E.

Correctness of the simulation.

Let $\mathsf{Real}(x, w)$ be the view of V^* in a real interaction $\langle \mathsf{P}(w) \leftrightarrows \mathsf{V}^* \rangle(x)$, and let $\mathcal{S}(x)$ be the view generated by our simulation procedure. Let Bad be the event over the coins of $\mathcal{S}$, that the simulation gets stuck; namely, it reaches a proof in some session in which the unlearnable secret was not yet extracted. Since the only difference between the experiments Real and $\mathcal{S}$ is the witness used to simulate the proof, it follows from the WI property of rsWIAOK that

$$\{\mathsf{Real}(x, w)\}_{(x,w)\in\mathcal{R}_\mathcal{L}} \approx_c \{\mathcal{S}(x)|\neg\mathsf{Bad}\}_{x\in\mathcal{L}} \ .$$

It is thus enough to show that

$$\Pr_{\mathcal{S}(x)}[\mathsf{Bad}] = \mathrm{negl}(n) \ ,$$

where $n = |x|$. For this purpose, we need to show that, in every session, one of the extraction attempts succeeds. The difficulty is that the circuit we are extracting from is performing simulation recursively, and this simulation may also get stuck and result in one of circuits (created by EMULATE along the main thread) not computing the function correctly. To bound the probability of such an event, we consider the following experiment: in the experiment, before executing the extractor on some circuit, we execute this circuit several time (using independent randomness) to see is the execution of this circuit is likely to get stuck. The simulation in the circuit will recursively follow the same augmented behavior. If in any of the executions the circuit gets stuck, we do not even try to extract just admit failure. We show that the probability of getting stuck, in this modified experiment, is still negligible. Intuitively, this guarantees that no attempt to extract, in the real simulation, will ever fail due to a recursive simulation getting stuck (except with negligible probability). We next define this experiment in more detail.

Let $\mathcal{S}'$ be the experiment that is defined like $\mathcal{S}$ except that all the calls to the function EXTRACT are replaced by calls to a new function FORK. The function FORK starts off just like the function EXTRACT; that is, on parameters (T, m), FORK calls the function EMULATE with the same parameters, obtains a circuit C, and executes the extractor E of the unobfuscatable functions family on C. FORK then repeats the following n independent times: sample $q \leftarrow \mathcal{D}(1^n)$ and run $C(q)$. We shall refer to the computation transcript produced by C in the i-th execution (out of n executions) as the i-th simulation performed by FORK.

After every such simulation performed by FORK, the simulation is rewound back to the state T. If any of the simulations by FORK gets stuck, we say that FORK gets stuck as well, and this failure propagates up. More precisely:

- If the function MAIN makes a call to FORK that gets stuck, the entire simulation gets stuck.
- If FORK makes a recursive call to FORK that gets stuck, the outer execution of FORK also gets stuck.
- If a circuit C constructed by EMULATE makes a call to FORK that gets stuck, C outputs $\perp$.

Finally, FORK outputs the same as E on C (similarly to the function EXTRACT).

Clearly

$$\Pr_{\mathcal{S}'(x)}[\mathsf{Bad}] \geq \Pr_{\mathcal{S}(x)}[\mathsf{Bad}] \ ,$$

and therefore it is enough to show that

$$\Pr_{\mathcal{S}'(x)}[\mathsf{Bad}] = \mathrm{negl}(n) \ .$$

Assume towards contradiction that there exist a polynomial p such that, for infinitely many values of $x \in \mathcal{L} \cap \{0,1\}^n$, $\Pr_{\mathcal{S}'(x)}[\mathsf{Bad}] \geq p(n)$, and let us fix any such x. Whenever Bad occurs in $\mathcal{S}'$, the simulation performed by MAIN or by some call to FORK reaches the proof in some session while the unlearnable secret was not previously extracted. We look at the execution "thread" that contains the entire simulated interaction from the beginning of the experiment up until the point where the simulation gets stuck. (Note that this thread might spread across several nested calls to FORK.) We can identify every thread of execution by the number of previous calls to FORK and the index of the simulation in the current call to FORK for every level of the recursion.

Let $\mathsf{Bad}_1^{t,s}$ be the event where the execution gets stuck in thread t and session s. Since the total number of threads and sessions is polynomial, there exist a thread t and a session s such that the event $\mathsf{Bad}_1 = \mathsf{Bad}_1^{t,s}$ occurs with some polynomial probability $p_1(|x|)$.

To argue that some of the slots of session s in thread t are "light", that is, they do not contain too many slots of other sessions, we will focus on a single level of the recursion that contains many of the slots of the session. Let Bad_2^l be the event that Bad_1 occurs and the call to FORK, in recursion level l and thread t, contains at least 2Δ full slots of session s. Since the simulation in the thread t must contains all n^δ slots of session s, in order to get stuck in s, and since the maximal number of nested calls to FORK is bounded by $d = \frac{c'}{\delta'} = O(1)$, it follows that there exist a level l such that the event $\mathsf{Bad}_2 = \mathsf{Bad}_2^l$ occurs with some polynomial probability $p_2(|x|)$. Let m_l be the parameter of FORK in recursion level l.

Since the simulation of FORK in thread t in level l of the recursion contains a bounded amount of slots from all sessions, but many slots of session s, there must be many slots of session s that are concurrent to a relatively small number of other slots. Intuitively, this means that the extraction from these light slots is more likely to succeed. Let S_i be the random variable representing the simulation of the i-th slot of session s that starts on recursion level l, in thread t (if no such slot exist S_i is empty). Let G_i be the event that S_i is not empty, contains no aborts, and the number of other slots that start concurrently to S_i is at most $\frac{m_l}{\Delta}$. Recall that before the simulation of a slot on recursion level l, $\mathcal{S}'$ makes a recursive call to FORK with parameter $\frac{m_l}{\Delta}$. Since the total number of sessions that start in the function FORK in level l is at most m_l, we have that whenever Bad_2 occurs, $|\{i \in [2\Delta] | G_i\}| \geq \Delta$.

Since with noticeable probability level l of thread t has many light slots, we expect that in at least one of these slots, if we rewind the simulation to the beginning of the slot, and simulate it again with independent randomness it will remain light with some constant probability. We will show that the extraction from such slot is likely to succeed. Let H_i be the event that $\Pr[G_i | S_1, \ldots, S_{i-1}] < 1/8$. Let Bad_3^i be that event $\mathsf{Bad}_2 \wedge \neg H_i$. The following lemma, together with the fact that $\Pr_{\mathcal{S}'(x)}[\mathsf{Bad}_2] = p_2(n)$, implies that there exist $i^* \in [2\Delta]$ such that event $\mathsf{Bad}_3 = \mathsf{Bad}_3^{i^*}$ occurs with some polynomial probability $p_3(|x|)$.

Lemma 5.4. $\Pr[\{(|\{i \in [2\Delta] | G_i\}| \geq \Delta\} \wedge \{H_1 \wedge \cdots \wedge H_{2\Delta}\}] \leq \mathrm{negl}(n)$

Proof. First we show that for every set $\{i_j\}_{j \in [\Delta]} \subseteq [2\Delta]$ and for every $0 \leq d \leq \Delta$

$$\Pr\left[\left(\bigwedge_{j=1+\Delta-d}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=1+\Delta-d}^{\Delta} H_{i_j}\right) \;\middle|\; \left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{\Delta-d} H_{i_j}\right)\right] \leq 8^{-d}$$

For $d = 0$ the claim clearly holds. Assuming the claim holds for $d-1$ we have:

$$
\begin{aligned}
&\Pr\left[\left(\bigwedge_{j=1+\Delta-d}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=1+\Delta-d}^{\Delta} H_{i_j}\right) \;\middle|\; \left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{\Delta-d} H_{i_j}\right)\right] \\
&\leq \Pr\left[\left(\bigwedge_{j=1+\Delta-d}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=2+\Delta-d}^{\Delta} H_{i_j}\right) \;\middle|\; \left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{1+\Delta-d} H_{i_j}\right)\right] \\
&\leq \Pr\left[G_{i_{1+\Delta-d}} \;\middle|\; \left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{1+\Delta-d} H_{i_j}\right)\right] \\
&\cdot \Pr\left[\left(\bigwedge_{j=2+\Delta-d}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=2+\Delta-d}^{\Delta} H_{i_j}\right) \;\middle|\; \left(\bigwedge_{j=1}^{1+\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{1+\Delta-d} H_{i_j}\right)\right] \\
&\leq \Pr\left[G_{i_{1+\Delta-d}} \;\middle|\; \left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{1+\Delta-d} H_{i_j}\right)\right] \cdot 8^{1-d} \\
&\leq 8^{-d} \ ,
\end{aligned}
$$

where the last inequality is due to the fact that the event $\left(\bigwedge_{j=1}^{\Delta-d} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{1+\Delta-d} H_{i_j}\right)$ is contained in the event $H_{i_{1+\Delta-d}}$. By setting $d = \Delta$ we get that

$$
\Pr\left[\left(\bigwedge_{j=1}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{2\Delta} H_j\right)\right] \leq \Pr\left[\left(\bigwedge_{j=1}^{\Delta} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{\Delta} H_{i_j}\right)\right] \leq 8^{-\Delta} \ .
$$

Finally, there are $\binom{2\Delta}{\Delta} < 2^{2\Delta}$ sets $\{i_j\}_{j\in[\Delta]} \subseteq [2\Delta]$ of size Δ and therefore:

$$
\Pr\left[\{|\{i \in [2\Delta]|G_i\}| \geq \Delta\} \wedge \{H_1 \wedge \cdots \wedge H_{2\Delta}\}\right] \leq \left(\frac{4}{8}\right)^{\Delta} \leq \mathrm{negl}(n) \ .
$$

□

Let T be the state of the simulation just before the beginning of the i^*-th slot. Let good be the event that the i^*-th slot exists, and the probability over the rest of the experiment, starting from T, that Bad_3 occurs is at least $p_3(n)/2$. Since $\Pr_{\mathcal{S}'(x)}[\mathsf{Bad}_3] = p_3(n)$, we have that $\Pr_{\mathcal{S}'(x)}[\mathsf{good}] \geq p_3(n)/2$. Now, conditioned on a good prefix T of the execution, $\Pr_{S_i}[G_i \mid T] \geq 1/8 = 2/16$ (this will be useful in a few lines).

Before the simulation of the i^*-th slot the function FORK is called, and tries to extract the unlearnable secret of the session s. Let S'_{i^*} be a random variable representing a simulation attempt made by FORK (out of the n identically distributed attempts). S'_{i^*} is distributed like S_{i^*} except that:

- S'_{i^*} never contains more than $\frac{m_l}{\Delta}$ other slots, while S_i may contain up to m_l concurrent slots. However, this difference does not hold conditioned on the event G_{i^*}.
- The simulation of S'_{i^*} makes recursive calls to FORK with parameter $\frac{m_l}{\Delta^2}$, while the simulation of S_{i^*} calls FORK with parameter $\frac{m_l}{\Delta}$; this means that the simulation of S_{i^*} is more likely to extract a witness, and less likely to get stuck. In particular, note that the above difference no longer holds conditioned on the event that the simulation of S'_{i^*} does not get stuck. We denote this event by NS.

Since G_{i^*} implies that S_{i^*} contains the end of the i^*-th slot, and that the messages (q, a) of the i^*-th slot satisfy $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1$, we have that $\mathsf{NS} \wedge G_{i^*}$ implies that the same holds for S'_{i^*}. Note that NS must occur with probability at least $15/16$; otherwise, one of the n simulation attempts made by FORK, starting from T, will get stuck with overwhelming probability and therefore Bad_2 will only occur with negligible probability; however, this will contradict our choice of T (recall that Bad_2 occurs only when the simulation gets stuck in thread t and not in the recursive call to FORK after state T). We thus have that $\Pr[\mathsf{NS} \wedge G_{i^*} \mid T] \geq 1/16$ and therefore, with probability at least $1/16$, S'_{i^*} contains the end of the i^*-th slot, and an execution of $C(q)$, performed by FORK starting from state T, outputs a such that $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a) = 1$ with probability at least $1/16$. By the non-black-box learnability property of the unobfuscatable function, when FORK invokes E on C with $\epsilon = \frac{1}{16}$, it outputs z such that $(\mathsf{vk}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}}$ with overwhelming probability. However, by the choice of T, the probability that Bad_3 occurs and session s is not solved on thread t must be noticeable.

Simulation running time. Let $T_{\text{MAIN}}(m), T_{\text{EMULATE}}(m), T_{\text{EXTRACT}}(m)$ be the running times of the functions MAIN, EMULATE, EXTRACT, given parameter m. One can verify that following relations hold:

$$\begin{aligned} T_{\text{MAIN}}(m) &< p_1(n) \cdot T_{\text{EXTRACT}}(\frac{m}{\Delta}) + p_2(n) \\ T_{\text{EXTRACT}}(m) &< p_3(T_{\text{EMULATE}}(m)) \\ T_{\text{EMULATE}}(m) &< p_4(T_{\text{MAIN}}(m)) \end{aligned}$$

And therefore

$$T_{\text{MAIN}}(m) < p_1(n) \cdot p_5(T_{\text{MAIN}}(\frac{m}{\Delta})) + p_2(n)$$

Where p_1 to p_5 are polynomials that depend only on V^*. Specifically, note that p_3 is such a polynomial since EXTRACT only runs E with a constant value of ϵ. For $m \leq 1$ the MAIN stops whenever a new slot starts and does not make any calls to EXTRACT. Therefore, $T_{\text{MAIN}}(m) < p_0(n)$ where p_0 is a polynomial that depend only on V^*. We get that:

$$T_{\text{MAIN}}(m) < p^d(n)$$

where p is a polynomial that depends only on V^* and $d = \lceil \log_{\Delta} m \rceil + 1$. Since $\Delta = n^{\delta'}$ and the main simulation executes MAIN with parameter $m = n^{c'}$ we have $d = O(1)$ and the simulation running time is polynomial.

This concludes the proof of Lemma 5.3 showing that Protocol 1 is concurrent zero-knowledge.

□

5.4 Resettably-sound ZK from Minimal Assumptions

In this section, we show how to construct resettably-sound ZK, and resettably-sound concurrent ZK protocols based only on one-way functions, by directly using robust unobfuscatable function, instead of (ZAP-based) verifiable robust ones. As a corollary, we also get a resettable ZK argument of knowledge based on one-way function, by plugging our protocol into the transformation of [BGGL01].

Let $\mathcal{G}$ be a family of robust unobfuscatable functions with respect to a hardcore family $\{\mathcal{HC}_n\}$ (as defined in Definition 2.5, and constructed in Sections 2 and 3 based on one-way functions). Let φ be a one-way function. We make use of a 2-message statistically binding commitment Com (e.g., [Nao91]). Given a first receiver message r, we denote by Com_r the function that computes the sender's commitment message. Additionally, we use an instance-dependent resettable witness-indistinguishable argument rWI and resettably-sound witness-indistinguishable arguments of knowledge rsWIAOK (see Definition 5.1). The protocol closely follows the idea of making robust unobfuscatable functions (with a

hardcore secret) verifiable, as presented in Section 4.1. Non-interactive commitments are replaced with 2-message commitments. ZAPs are replaced with instance-dependent rWIs (as defined in Section 5.1.1), where the dependance is on the proven statement x. (The latter will be used by the verifier to prove that it acts properly, so they need to be sound only when $x \in \mathcal{L}$, and rWI only when $x \notin \mathcal{L}$). The protocol is described in Figure 2, where m is a parameter that controls the number of slots. similarly to Protocol 1 we set $m = 2$ to get constant round resettably-sound ZK and $m = n^{\Omega(1)}$ to get a resettably-sound concurrent ZK.

Protocol 2

Common Input: $x \in \mathcal{L} \cap \{0,1\}^n$.

Auxiliary Input to P**:** $w \in \mathcal{R}_{\mathcal{L}}(x)$.

1. P samples the first message r for a statistically binding commitment scheme and sends r to P.
2. V samples a bit $b \in \{0,1\}$.
   For $i \in \{0,1\}$, V samples $g_{\mathsf{k}_i} \in \mathcal{G}$ and $\mathsf{vk}_i = (h_i, y_i)$, where $h_i \leftarrow \mathcal{HC}_n$ is randomly chosen hardcore function, and $y_i = \varphi(h(\mathsf{k}_i))$.
   V computes $\mathsf{C} = \mathsf{Com}_r(b)$,$\mathsf{C}_i = \mathsf{Com}_r(\mathsf{k}_i)$ and sends $(\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$ to P.
3. Repeat the following function evaluation slot m times:
   (a) P samples $q \leftarrow \mathcal{D}(1^n)$ and sends q to V.
   (b) V evaluates the functions $a_0 = g_{\mathsf{k}_0}(q), a_1 = g_{\mathsf{k}_1}(q)$ and sends a_0, a_1 to P.
   (c) V proves to P using an x-dependent rWI^x argument that the following statement is correct:
   $$\left\{\{\mathsf{C} = \mathsf{Com}_r(0)\} \vee \left\{ \begin{array}{c} \mathsf{C}_0 = \mathsf{Com}_r(\mathsf{k}_0) \\ a_0 = g_{\mathsf{k}_0}(q) \end{array} \right\}\right\} \bigwedge \left\{\{\mathsf{C} = \mathsf{Com}_r(1)\} \vee \left\{ \begin{array}{c} \mathsf{C}_1 = \mathsf{Com}_r(\mathsf{k}_1) \\ a_1 = g_{\mathsf{k}_1}(q) \end{array} \right\}\right\}$$
4. P proves the following statement to V using an rsWIAOK:
   "$x \in \mathcal{L}$ or $\{\exists \mathsf{z} : y_0 = \varphi(\mathsf{z}) \vee y_1 = \varphi(\mathsf{z})\}$".

Figure 2: A resettably-sound (concurrent) ZK protocol from any one-way function

Theorem 5.1 (resettably-sound (concurrent) ZK from one-way functions)**.** *Protocol 2 with $m = 2$ is a constant round resettably-sound ZK protocol, and with $m = n^{\Omega(1)}$, it is a resettably-sound concurrent ZK protocol.*

Corollary 5.3 (of Theorem 5.1 and [BGGL01])**.** *There exist a resettable ZK argument of knowledge based on one-way functions.*

The proof of Theorem 5.1 closely follows the proofs of Lemmas 5.3,5.1, and 5.2, and is omitted. Corollary 5.3 follows by plugging in our one-way function based resettably-sound ZK protocol into the transformation of [BGGL01].

5.5 A 3-Message Simultaneous Resettable WI Argument of Knowledge

In this section, we construct a 3-message simultaneously-resettable WI proof-of-knowledge protocol based on robust unobfuscatable functions, where knowledge extraction is performed by a non-black-box extractor. As in Section 4.1, our protocol will use the idea of turning a single witness statement into a two independent-witnesses statement as done in [FS90, COSV12].

Lemma 5.5. *Protocol 3 is a resettable WI argument of knowledge.*

Protocol 3

Common Input: $x \in \mathcal{L} \cap \{0,1\}^n$.

Auxiliary Input to P: $w \in \mathcal{R}_{\mathcal{L}}(x)$.

1. P performs the following:
   - samples two keys $\mathsf{k}_0, \mathsf{k}_1 \leftarrow \{0,1\}^n$ for $f_{\mathsf{k}_0}, f_{\mathsf{k}_1} \in \mathcal{F}$, and a bit $b \leftarrow \{0,1\}$.
   - samples two hardcore functions $h_0, h_1 \leftarrow \mathcal{HC}_n$ and computes $r_0 = h_0(\mathsf{k}_0), r_1 = h_1(\mathsf{k}_1)$.
   - sends V: $\mathsf{C}_0 = \mathsf{Com}(\mathsf{k}_0), \mathsf{C}_1 = \mathsf{Com}(\mathsf{k}_1), \mathsf{C} = \mathsf{Com}(b)$, and $h_0, \mathsf{e}_0 = w \oplus r_0, h_1, \mathsf{e}_1 = w \oplus r_1$.
2. V performs the following:
   - samples randomness $r \leftarrow \{0,1\}^{\mathrm{poly}(n)}$ for an rZAP, and an input $q \leftarrow \mathcal{D}(1^n)$.
   - sends (q, r) to P.
3. P performs the following:
   - computes $a_0 = f_{\mathsf{k}_0}(q), a_1 = f_{\mathsf{k}_1}(q)$.
   - computes an rZAP proof π for the statement:

$$\left\{ \{\mathsf{C} = \mathsf{Com}(0)\} \vee \left\{ \begin{array}{c} \mathsf{C}_0 = \mathsf{Com}(\mathsf{k}_0) \\ a_0 = g_{\mathsf{k}_0}(q) \\ w \in \mathcal{R}_{\mathcal{L}}(x) \\ \mathsf{e}_0 = w \oplus h_0(\mathsf{k}_0) \end{array} \right\} \right\} \bigwedge \left\{ \{\mathsf{C} = \mathsf{Com}(1)\} \vee \left\{ \begin{array}{c} \mathsf{C}_1 = \mathsf{Com}(\mathsf{k}_1) \\ a_1 = g_{\mathsf{k}_1}(q) \\ w \in \mathcal{R}_{\mathcal{L}}(x) \\ \mathsf{e}_1 = w \oplus h_1(\mathsf{k}_1) \end{array} \right\} \right\}$$

   - sends V: a_0, a_1, π.
4. V verifies the rZAP proof π, and decided whether to accept accordingly.

Figure 3: An rWI 3-message Argument of Knowledge (implying srWIAOK)

Since the protocol is a 3-message protocol, we can apply the [BGGL01] transformation, where V derives its randomness by applying a PRF to the transcript; as a corollary, we get the following theorem:

Corollary 5.4. *Assuming trapdoor permutations exist, there exist a 3-message simultaneously-resettable WI argument of knowledge (with non-black-box knowledge extraction).*

We now give a proof sketch of the lemma.

Proof sketch of Lemma 5.5. We start by showing that the protocol is resettably WI. Let

$$(\mathcal{X}, \mathcal{W}_0, \mathcal{W}_1) = \{(x, w_0, w_1) : (x, w_0), (x, w_1) \in \mathcal{R}_{\mathcal{L}}\}$$

be any infinite sequence of instances in $\mathcal{L}$ and corresponding witness pairs. We next consider a sequence of hybrids starting with an hybrid describing an interaction with a prover that uses $w_0 \in \mathcal{W}_0$, and ending with an hybrid describing an interaction with a prover that uses $w_1 \in \mathcal{W}_1$, where both w_0, w_1, are witness for some $x \in \mathcal{X}$. We shall prove that no efficient verifier can distinguish between any two hybrids in the sequence. The list of hybrids is given in Table 1. The can be though of as two sequences. One 0.1-6, starts from witness w_0, and the other 1.1-6 starts at witness w_1. We will show that within these sequences the hybrids are indistinguishable, and then will show that 0.6 is indistinguishable from 1.6.

Hybrid 0.1: describes a true interaction of a resetting verifier V^* with an honest prover P that uses w_0 as a witness for the statement $x \in \mathcal{L}$. In particular, the rZAP uses the witness $((\mathsf{k}_0, w_0), (\mathsf{k}_1, w_0))$

hyb	zapw_b	C_b	r_b	$\mathsf{e}_b \oplus r_b$	zapw_{1-b}	C_{1-b}	r_{1-b}	$\mathsf{e}_{1-b} \oplus r_{1-b}$
0.1	(k_b, w_0)	k_b	$h_b(\mathsf{k}_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
0.2	b	k_b	$h_b(\mathsf{k}_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
0.3	b	$0^{\lvert\mathsf{k}_b\rvert}$	$h_b(\mathsf{k}_b)$	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
0.4	b	$0^{\lvert\mathsf{k}_b\rvert}$	u	w_0	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
0.5	b	$0^{\lvert\mathsf{k}_b\rvert}$	u	w_1	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
0.6	(k_b, w_1)	k_b	$h_b(\mathsf{k}_b)$	w_1	(k_{1-b}, w_0)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_0
1.6	(k_b, w_0)	k_b	$h_b(\mathsf{k}_b)$	w_0	(k_{1-b}, w_1)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_1
1.2-5	...	...	...	...	...	...	...	...
1.1	(k_b, w_1)	k_b	$h_b(\mathsf{k}_b)$	w_1	(k_{1-b}, w_1)	k_{1-b}	$h_{1-b}(\mathsf{k}_{1-b})$	w_1

Table 1: The sequence of hybrids; the bit b corresponds the bit commitment C; the blue cells are those different comparing to the previous hybrid.

(the witness also includes the randomness for the commitments C_0 and C_1, but for notational brevity, shall omit it.) In Table 1, the witness used in part 0 of the rZAP is referred to as zapw_0, and the one corresponding to 1 in zapw_1.

Hybrid 0.2**:** This hybrid differs from the previous only the witness used in any rZAP. Specifically, for the bit b given by C, the witness for the rZAP is set to be $(b, (\mathsf{k}_{1-b}, w_0))$, instead of $((\mathsf{k}_b, w_0), (\mathsf{k}_{1-b}, w_0))$. (Again the witness should include the randomness for the commitment C, and C_{1-b}, but is omitted from our notation.) Since the rZAP is resettably WI, this hybrid is computationally indistinguishable from the previous one.

Hybrid 0.3**:** In this hybrid, the commitment C_b is given to $0^{|\mathsf{k}_b|}$, instead of to k_b. This hybrid is computationally indistinguishable from the previous one due to the computational hiding of the commitment scheme C.

Hybrid 0.4**:** In this hybrid, instead of sending the verifier the hardcore secret $h_b(\mathsf{k}_b)$, it is given a random independent string $u \leftarrow \{0,1\}^{|h_b(\mathsf{k}_b)|}$. Computational indistinguishability of this hybrid from the previous one, follows by the black-box indistinguishability of $\mathcal{F}$ (Definition 2.5). Indeed, note that any distinguisher here can be turned into a distinguisher against $\mathcal{F}$ and its corresponding hardcore family $\mathcal{HC}$, by treating the oracle as k_b and simulating all the other elements in the experiment.

Hybrid 0.5**:** In this hybrid, the padded value e_b is taken to be $w_1 \oplus r_b$, instead of $w_0 + \oplus r_b$. Since r_b is now uniform and independent of all other elements, this hybrid induces the exact same distribution as the previous hybrid.

Hybrid 0.6**:** This hybrid now backtracks, returning to the same experiment as in hybrid 0.1 with the exception that the rZAP witness is now $((\mathsf{k}_b, w_1), (\mathsf{k}_{1-b}, w_0))$ instead of $((\mathsf{k}_b, w_0), (\mathsf{k}_{1-b}, w_0))$. This indistinguishability follows exactly as when moving from 0.1 to 0.5 (only backwards).

Hybrids 1.1 **to** 1.6**:** These hybrids are symmetric to the above hybrids, only that they start from w_1 instead of w_0. This means that they end in 1.6 which uses an rZAP witness $((\mathsf{k}_b, w_0), (\mathsf{k}_{1-b}, w_1))$, which is the same as 0.6, only in reverse order.

Hybrids 0.6 **and** 1.6 **are computationally indistinguishable.** This follows directly from the computationally hiding of $\mathsf{C} = \mathsf{Com}(b)$. Indeed, assume towards contradiction that V distinguishes the two hybrids. Concretely, denote the probability it outputs 1 on 0.6 by p_0, and the probability it outputs 1 on 1.6 by p_1, and assume WLOG that $p_0 - p_1 \geq \epsilon$, for some noticeable $\epsilon = \epsilon(n)$. We can construct a predictor that given a commitment $\mathsf{C} = \mathsf{Com}(b)$ to a random bit $b \leftarrow \{0,1\}$, guesses b with probability $\frac{1+\epsilon}{2}$. The predictor, samples a random $b' \leftarrow \{0,1\}$ as a candidate guess for b, and performs the experiment corresponding to 0.6, only that it locates w_0 and w_1 according to b', rather than the unknown b. If

the distinguisher outputs 1, the predictor guesses $b = b'$ and otherwise it guesses $b = 1 - b'$.

Conditioned on $b = b'$, V is experiencing 0.6, and thus the guess will be correct with probability p_0; conditioned on $b = 1 - b'$, V is experiencing 1.6, and the guess will be right with probability $1 - p_1$. So overall the guessing probability is $\frac{p_0}{2} + \frac{1-p_1}{2} \geq \frac{1}{2} + \frac{\epsilon}{2}$. This completes the proof that the protocol is resettable WI.

It is left to show that the protocol is an argument of knowledge. Indeed, let P^* be any prover that convinces the honest verifier of accepting with noticeable probability $\epsilon = \epsilon(n)$, then with probability at least $\epsilon/2$ over its first message, it holds with probability at least $\epsilon/2$ over the rest of the protocol that P^* convinces V. Let us call such a prefix good. Now for any good prefix, we can consider the perfectly binding induced commitment to the bit b, and from the soundness of the rZAP, we get a circuit that with probability at least $\epsilon/2$ computes correctly the function $f_{\mathsf{k}_{1-b}}$, and gives a valid witness $w \in \mathcal{R}_{\mathcal{L}}$, padded with $h_{1-b}(\mathsf{k}_{1-b})$. This in particular, means that we can first sample a prefix (hope it is good), and then invoke the non-black-box learnability guarantee (Definition 2.5) to learn $h_{1-b}(\mathsf{k}_{1-b})$, and thus also the witness w. (Since we do not know what is the bit b, we will need to construct to circuits for both options of b, and try to extract from both, just as was done in the proof of Claim 4.2.) This completes the proof of Lemma 5.5. □

Acknowledgements

We thank Ran Canetti for many enlightening discussions and comments. We thank Vipul Goyal for interesting discussions on constructing robust unobfuscatable functions. We thank Rafael Pass for sharing with us his own ideas on obtaining resettable-soundness from one-way functions. We also thank Rafael for pointing out a bug in an earlier version of this work. We are especially grateful to Amit Sahai for sharing with us his perspective on the question of robust unobfuscatable functions. In particular, while we were mainly interested in applications to resettable protocols, Amit pointed out to us the connection to the impossibility of approximate obfuscation raised in [BGI$^+$01]. Finally, we thank Leo Reyzin for valuable discussions and advice.

References

[AW07] Ben Adida and Douglas Wikström. How to shuffle in public. In *TCC*, pages 555–574, 2007.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In *CRYPTO*, pages 520–537, 2010.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[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 *CRYPTO*, pages 1–18, 2001.

[Blu81] Manuel Blum. Coin flipping by telephone. In *Proceedings of the 18th Annual International Cryptology Conference*, pages 11–15, 1981.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986.

[BP12] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, 2012.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010.

[COSV12] Chongwon Cho, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Simultaneously resettable arguments of knowledge. In *TCC*, pages 530–547, 2012.

[CPS12] Ka-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security, 2012. manuscript.

[Dam89] Ivan Damgård. On the existence of bit commitment schemes and zero-knowledge proofs. In *CRYPTO*, pages 17–27, 1989.

[DFG+11] Yi Deng, Dengguo Feng, Vipul Goyal, Dongdai Lin, Amit Sahai, and Moti Yung. Resettable cryptography in constant rounds - the case of zero knowledge. In *ASIACRYPT*, pages 390–406, 2011.

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.

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

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, New York, NY, USA, 1989. ACM.

[GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, pages 678–687, 2011.

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000.

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In *EUROCRYPT*, pages 54–71, 2009.

[HMLS07] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. In *TCC*, pages 214–232, 2007.

[HRSV07] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. In *TCC*, pages 233–252, 2007.

[MR01] Silvio Micali and Leonid Reyzin. Min-round resettable zero-knowledge in the public-key model. In *EUROCRYPT*, pages 373–393, 2001.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[OV12] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 2012.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.