ON NON-BLACK-BOX SIMULATION AND THE IMPOSSIBILITY OF APPROXIMATE OBFUSCATION *

NIR BITANSKY† AND OMER PANETH‡

Abstract. The introduction of a non-black-box simulation technique by Barak (FOCS 2001) has been a major landmark in cryptography, breaking the previous barriers of black-box impossibility. Barak's technique has given rise to various powerful applications and it is a key component in all known protocols with non-black-box simulation.

We present the first non-black-box simulation technique that does not rely on Barak's technique (or on non-standard assumptions). Invoking this technique, we obtain new and improved protocols resilient to various resetting attacks. These improvements include weaker computational assumptions and better round complexity.

A prominent feature of our technique is its compatibility with rewinding techniques from classic black-box zero-knowledge protocols. The combination of rewinding with non-black-box simulation has proven instrumental in coping with challenging goals as: simultaneously-resettable zero-knowledge, proofs of knowledge, and resettable-security from one-way functions. While previous works required tailored modifications to Barak's technique, we give a general recipe for combining our technique with rewinding. This yields simplified resettable protocols in the above settings, as well as improvements in round complexity and required computational assumptions.

The main ingredient in our technique is a new impossibility result for general program obfuscation. The results extend the impossibility result of Barak et al. (CRYPTO 2001) to the case of obfuscation with approximate functionality; thus, settling a question left open by Barak et al.. In the converse direction, we show a generic transformation from any resettably-sound zero-knowledge protocol to a family of functions that cannot be obfuscated.

*This paper is the full version of extended abstracts that appeared in FOCS 2012 [9] and STOC 2013 [10]

†MIT. Email: nirbitan@csail.mit.edu. This research was done while the first Author was at Tel Aviv University and supported by the Check Point Institute for Information Security, an ISF grant 20006317, the Fulbright program, and an IBM Ph.D. fellowship. Part of this research was done while visiting Boston University and IBM T. J. Watson Research Center.

‡Boston University. Email: omer@bu.edu. Supported by the Simons award for graduate students in theoretical computer science and NSF award 1218461.

1. Introduction. Zero-knowledge (ZK) protocols [31] are a cornerstone of modern cryptography; they can express all NP computations [30], and are essential to almost any form of secure computation. The ZK guarantee of a protocol is established by exhibiting an efficient *simulator* that can simulate the view of any malicious verifier from the verifier's code and the statement alone. Following the common practice of black-box reductions, the first ZK protocols all relied on simulators that only use the verifier as a black-box, without making any explicit use of its code. However, while sufficient for a variety of powerful applications, ZK protocols with black-box simulation were soon shown to have inherent limitations. A known example is the impossibility of constant-round public-coin ZK [28]. Surpassing such black-box impossibilities was considered to be an inconceivable task.

This barrier was crossed with the groundbreaking result of Barak [1] that introduced a non-black-box simulation technique. Barak's technique allowed, in particular, to achieve constant-round public-coin ZK. Subsequently, the technique was utilized to achieve various cryptographic goals, most of which were previously limited by black-box impossibilities, e.g., [4, 2, 6, 41, 42, 7, 43, 44, 20, 35, 33, 34, 45, 14, 32].

Today, an impressive variety of protocols rely on non-black-box simulation; however, the common base of all of these protocols is the very same technique of Barak. The main question addressed in this work is *whether Barak's technique is inherent in non-black-box simulation, or whether black-box impossibilities may be circumvented using different techniques and tools*. Beyond improving our understanding of non-black-box simulation, answering the above question may also lead to protocols with improved features.

Resettable soundness - the bottleneck in black-box impossibilities. To answer the above question, we focus on a specific setting that seems fundamental in ZK black-box impossibilities - the setting of *resettably-sound zero-knowledge* [4, 38]. Indeed, many of the known black-box ZK impossibilities can be derived by a reduction to the impossibility of resettably-sound ZK with a black-box simulator [28, 4, 46].

A ZK argument is resettably-sound if it remains sound, even if an adversarial prover may reset the honest verifier to its initial state and random tape, and repeat the interaction in any way it chooses (equivalently, the verifier may be rewound to any previous state). As observed by Barak et al. [4], in resettably-sound protocols, the ZK requirement cannot be fulfilled by a black-box simulator (except for trivial languages). Intuitively, the difficulty is that a cheating prover may execute a resetting attack that emulates the strategy of the black-box simulator, and thus violate soundness. In contrast, a resettably-sound ZK protocol was constructed by Barak et al. [4] using Barak's non-black-box simulation technique.

The difficulty in extending Baraks technique. Attempting to extend [4] to resist more elaborate types of resetting attacks (e.g., *simultaneous resetting*), one encounters significant technical difficulties. To overcome these difficulties, subsequent works have extended Baraks technique in various ways [20, 35, 34, 16, 14, 32, 19]. A common theme in these works is to extend Baraks technique so it can be combined with rewinding techniques. While allowing to leverage powerful black-box simulation techniques (e.g., [48]), they require tailored modifications to Barak's technique, and result in rather complicated protocols.

1.1. Our Contribution. In this work, we present a new non-black-box simulation technique that is fundamentally different from Barak's technique. Using our technique we give new protocols for resettably-sound zero-knowledge and related tasks.

The simulation in these protocols uses our new technique as well as rewinding techniques. In contrast to the tailored modifications of Baraks technique in previous work, we give a general recipe for combining our technique with rewinding. This yields simplified resettable protocols for various tasks, as well as improvements in round complexity and required computational assumptions.

Whereas a main technical tool in Barak's technique is PCP-based universal arguments, our technique does not invoke universal arguments in any way. In the heart of our technique, is a new method for extracting a short trapdoor from adversarial code. The extraction technique can be interpreted as a new impossibility result for general program obfuscation. The result extends the impossibility result of Barak et al. [5] to the case of obfuscation with approximate functionality; thus, settling a question left open by Barak at al..

Finally, we show that any resettably-sound ZK argument can be transformed into a family of functions that cannot be obfuscated, thereby establishing a two-way connection between the impossibility of obfuscation and non-black-box simulation.

1.2. Obfuscation and Non-Black-Box Simulation. 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 Hada [36] and Barak et al. [5], which formalized secure obfuscation according to the *virtual black-box* notion. At high-level, this notion requires that whatever an efficient learner can deduce, given an obfuscation $\tilde{P}$ of a program P, should also be learnable, given only black-box access to P. The same work shows, however, that for some programs, this notion is not achievable. Concretely, [5] show that there exists an *unobfuscatable* family of functions $\{f_{\mathsf{k}}\}$ for which *any* program $\tilde{P}$ that computes a function f_{k} leaks the key k; that is, k can be efficiently *extracted* from $\tilde{P}$'s code. However, a randomly chosen key k cannot be learned given only black-box access to f_{k}.

At an intuitive level, unobfuscatable functions suggest a meaningful way to use the code of the adversarial verifier in non-black-box simulation. Following this intuition we design a protocol where the verifier evaluates an unobfuscatable function f_{k}. A non-black-box simulator would be able to learn k from the code of any malicious verifier and use it to simulate. In contrast, a resetting prover, like a black-box simulator, would only get black-box access to the verifier and would not learn k.

A naive protocol and the problem of non-robust extraction. We now describe a naive resettably-sound ZK protocol based on the intuition presented above. Following the Feige-Lapidot-Shamir paradigm [22], the protocol proceeds in two phases: the first phase defines a trapdoor which is known by verifier but is "hard to compute" for the prover, and in the second phase, the prover gives a witness indistinguishable (WI) proof that either the statement is correct or that it "knows" the trapdoor. In our case, to obtain resettable-soundness we will have to ensure that the trapdoor is hard to compute, even given the ability to reset the verifier. We will also use a resettably-sound WI proof in the second phase of the protocol (such proofs follow directly from classic protocols [24]). For ZK, we will exhibit a simulator that, given the code of the verifier, can extract the trapdoor and successfully complete the simulation.

As suggested above, the trapdoor in our protocol will be a key k for an unobfuscatable function f_{k} chosen at random by the verifier. In the trapdoor generation phase of the protocol, the prover asks the verifier to evaluate the function f_{k} on an input x of his choice (for simplicity of exposition, we assume that this interaction indeed determines the key k and the trapdoor statement for the second phase). To show that the protocol is resettably-sound, we note that any prover that can reset the verifier in the first phase can at most evaluate the function f_{k} on many inputs x of his choice; however, given such "black-box" access to f_{k}, it is impossible to learn k and cheat in the second phase. To show that the protocol is ZK, we need to show a simulator that can extract the trapdoor k from the code of any cheating verifier. Assuming that the cheating verifier correctly evaluates the function f_{k} on any input sent by the prover, the simulator would be able to learn k from the code of the verifier's next message function. However, if on certain inputs x the verifier deviates from the protocol and does not respond with $f_{\mathsf{k}}(x)$, the extraction guarantee no longer holds and simulation cannot be finished.

1.3. Robust Unobfuscatable Functions and Approximate Obfuscation. A first solution to the above problem [9] is to evaluate the function f_{k} in the first phase of the protocol using a *secure function evaluation* protocol that secure against a resetting provers. Intuitively, in this solution, the secure function evaluation protocol guarantees that the verifier must correctly compute the function on all inputs, or always abort the computation. The solution in [9] relies on semi-honest *oblivious transfer*. Securing the secure function evaluation protocol against a resetting prover results in a rather complicated protocol with a large (but constant) number of rounds.

In this paper, we focus on a different approach relying on stronger unobfuscatable functions that provide a robust extraction guarantee [10] Specifically, it is possible to extract the key k from any program $\tilde{P}$ that *approximately* computes the function f_{k}, rather than exactly. More concretely, we define (publicly) verifiable *robust* unobfuscatable functions that have, in addition to the secret key k, an associated public verification key vk. The verification key can be used to verify that a pair (x, y) indeed satisfies $y = f_{\mathsf{k}}(x)$. On one hand, we require that a randomly chosen key k cannot be learned given the verification key vk and black-box access to f_{k}. On the other hand, it is possible to extract k from vk and any program $\tilde{P}$ that computes a function f_{k}, *even on some small (but noticeable) fraction of its inputs*. We present a construction of verifiable robust unobfuscatable functions based on ZAPs [21] and non-interactive commitments.

Using verifiable robust unobfuscatable functions fixes the naive protocol above. As before, the trapdoor will be a key k for a robust unobfuscatable function f_{k} chosen at random by the verifier. In the first phase of the protocol, the prover will receive the public verification key vk, and ask the verifier to evaluate the associated function on a *uniformly random* input x. The prover will use vk to verify that f_{k} was evaluated correctly and abort otherwise. We argue that the revised protocol is ZK: given the code of any cheating verifier, the verifier either makes the honest prover abort with overwhelming probability, in which case simulation is trivial, or the verifier evaluates f_{k} correctly on some noticeable fraction of inputs. In this case, we can extract k from the verifier's next message function and simulate the second phase.

Impossibility of approximate obfuscation. In addition to their application to resettably-sound ZK, verifiable robust unobfuscatable functions also settle an open question raised by Barak et al. [5] regarding the possibility of *approximate obfuscation*. In approximate obfuscation [5], the obfuscated program $\tilde{P}$ is only required to agree with P on some significant fraction of inputs. Through verifiable robust unobfuscatable functions, we give a negative answer to this question.

Constructing robust unobfuscatable functions. In this work, we focus on constructing robust unobfuscatable functions from minimal computational assumptions. Our construction consists of two steps: First, we construct a weaker object that we call (plain) *robust unobfuscatable functions*. Such functions have no verification key, however, they still allow extraction from any program $\tilde{P}$ that only compute f_{k} on a small fraction of inputs, but only assuming that on all other inputs $\tilde{P}$ outputs a special $\perp$ symbol. We construct such robust unobfuscatable functions from any one-way function. In the second step, we "compile" robust unobfuscatable functions into full fledged verifiable robust unobfuscatable functions based on ZAPs [21] and non-interactive commitments.

We note that a different construction, from stronger assumptions, can be obtained using ideas from [9]. The construction transforms any (non-robust) unobfuscatable function (such as the one of [5]) to a verifiable robust unobfuscatable functions, using a two-message *secure function evaluation* protocol against malicious parties [1].

[1] Since such protocols require super-polynomial simulation, the unobfuscatable function should also be unlearn-

Simulation from robust unobfuscatable functions vs. Barak's simulation. Both our protocol and Barak's protocol follow the Feige-Lapidot-Shamir paradigm [22], in which the prover ultimately proves a statement that has a trapdoor witness. A key technical difference between our technique and that of Barak is that our simulator extracts from the verifier's code a "short" trapdoor of some fixed polynomial length, whereas the simulation of Barak can be seen as using the entire verifier's code as a "long" trapdoor whose length is not a priori bounded by any fixed polynomial. To facilitate the use of such a long trapdoor, Barak's protocol involves a WI *universal argument* [3] while our protocols use a standard WI argument. The ability to extract a short trapdoor explains the compatibility between our technique and rewinding techniques, and enables our simplified constructions. Another difference between our technique an Barak's is that our in our technique, the verifier makes inherent use of private coins to sample its trapdoor, a key of an unobfuscatable function.

1.4. Resettable Protocols from Robust Unobfuscatable Functions. We next give an overview of the different resettably-secure protocols constructed based on our new technique. As we shall see, compered to known constructions, our protocols require less rounds, weaker assumptions, and are arguably simpler.

All of our protocols follow a similar design paradigm of *resettable slots* that abstracts the use of robust unobfuscatable functions. Commonly used in classic ZK protocols (with black-box simulation), a *slot* is a sub-protocol where the verifier proves knowledge of a trapdoor. While a cheating prover cannot learn the trapdoor from the slot's execution, the simulator extract the trapdoor by rewinding the verifier during the slot's execution. The difficulty in using slots within a resettably-sound ZK protocol is that a cheating prover can use resetting to extract the trapdoor and break soundness. The concept of a resettable slot, suggested by [20], requires that the trapdoor cannot be learned by rewinding the slot, but can still be extracted by the non-black-box simulator from the code of a verifier that executes the slot.

Concretely, in our resettable slot, the verifier's trapdoor is a key k for a verifiable robust unobfuscatable function, which is determined prior to the slot's execution by sending the verification key vk to the prover. The slot consists of the verifier evaluating f_{k} on a random input selected by the prover.

Following the resettable slot paradigm, we are able to take ZK protocols with black-box simulation and "add resettable-soundness" in a modular way, replacing (standard) slots with resettable slots. This results in protocols that are conceptually and practically simpler than those based on Barak's technique [20, 16, 14, 19].

Resettably-sound ZK in 4 messages. Plugging a resettable slot into the the 4-message ZK protocol of Feige and Shamir [23] results in a 4-message resettably-sound ZK protocol. Previous constructions relying on Barak's technique [4] required at least 6 messages.

Simulating such a protocol with a single slot requires verifiable robust unobfuscatable functions that have a fast enough extraction algorithm. We show how to construct such functions based on *rerandomizable fully homomorphic encryption*. Subsequently to our work, Chung et al. [17] constructed a 4-message resettably-sound ZK protocol based on one-way functions, relying on a different approach.

Resettably-sound ZK from one-way functions. By relying directly on robust unobfuscatable functions (that are not verifiable), we achieve resettably-sound ZK from the minimal assumption of one-way functions. Our protocol mimics the steps of compiling robust to verifiable robust unobfuscatable functions, but instead of using ZAPs, and non-interactive commitments, we use interactive (resettable) WI arguments and commitments from one-way functions. Our construction is an alternative to the first construction of resettably-sound

able against super-polynomial adversaries

ZK from one-way functions by Chung, Pass, and Seth [19] that showed how to replace the collision-resistent hash functions in to protocol of [4] with an interactive use of signatures.

Simultaneously-resettable ZK from one-way functions. The notion of resettable ZK [13] requires that even cheating verifiers that may reset the prover cannot gain knowledge from the protocol. Barak et al. [4] contemplated the existence of protocols that are simultaneously resettable ZK and resettably-sound. Such protocols were first constructed in the influential work of Deng, Goyal, and Sahai [20]. The main step in the construction of [20] is to construct a resettably-sound concurrent ZK protocol. To do so, they present a sophisticated extension of Barak's technique that carefully combines existing rewinding strategies from concurrent ZK.

We get a simpler construction of resettably-sound concurrent ZK by plugging our resettable slot into the concurrent ZK protocol of Richardson and Kilian [48]. Following the transformation of [20], we get a simultaneously-resettable ZK protocol. By directly using robust unobfuscatable functions (that are not verifiable), we get a resettably-sound concurrent ZK protocol from any one-way function where previous constructions also relied on one-to-one one-way function [20, 19]. Using a construction of simultaneously-resettable WI arguments from one way functions [18], within the transformation of [20] (as augmented by [18]), we get a construction of simultaneously-resettable ZK from one-way functions.

Our protocol inherits its polynomial number of rounds from the protocol of Richardson and Kilian. An open question is whether our technique can be combined with state of the art concurrent ZK protocols [37, 47] to reduce the number of rounds to poly-logarithmic. This turns out to depend on the relation between the unobfuscatable function's extraction time, and the depth of recursive rewinding in these protocols. Concretely, combining these protocols with our techniques requires linear time extraction. We do not know if such unobfuscatable functions exist.

Simultaneously-resettable WI argument-of-knowledge in 3 messages. The simultaneously-resettable ZK protocol of Deng,Goyal and Sahai [20] does not satisfy the desirable proof-of-knowledge property. Cho et al. [16] show a simultaneously-resettable WI argument-of-knowledge with a large constant number of rounds. This is in contrast to standard constructions of WI arguments-of-knowledge in 3 messages [24]. Using a resettable slot, we construct a 3-message simultaneously-resettable WI argument-of-knowledge. In our construction, the extraction property of the resettable slot is used to by the knowledge extractor to obtain a witness from the prover.

1.5. How to Construct Robust Unobfuscatable Functions, an Overview. The main technical challenge in this work is constructing (plain) robust unobfuscatable functions from one-way functions. We now give an overview of the main ideas on which our construction is based.

The starting point for our construction is the construction of Barak et al. [5] of (non-robust) unobfuscatable functions, and can be seen as a "random-self-reducible" version of it. We start by describing the construction of Barak et al., and then move on to describe the main modifications that we introduce.

The construction of Barak et al.. Barak et al. 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 explain how to modify the construction to rely on CCA-1 symmetric encryption (that can be based on one-way functions). For simplicity, we describe $f_{\mathsf{k}} = f_{a,b,\mathsf{sk}}$ as a randomized function; eventually, it is 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 any input that is an encryption of b, output b.

Given black-box access to a random function from the family, the only way to learn b is to first learn a, which will break the security of the encryption scheme, or to produce an encryption of b, which is information theoretically impossible, without first querying the function on 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. Then, we can learn b using another black-box application of C. Note that in the above construction we only learn b instead of the entire key of the function. Similarly, in the rest of our constructions we will only be able to extract some unlearnable property $\pi(\mathsf{k})$, rather than the entire key k of the unobfuscatable function; this will be sufficient for all of our applications.

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 errs on the single input a, we can no longer extract b. At high-level, we overcome this problem by defining a new function family $\mathcal{G}$ such that $\mathcal{F}$ is randomly reducible to $\mathcal{G}$. That is, for any input x, we can compute $f_{a,b,\mathsf{sk}}(x)$ by evaluating a corresponding $g \in \mathcal{G}$ on inputs that are individually random, but are correlated together according to x. It is important that the new function $\mathcal{G}$ remains black-box unlearnable as was $\mathcal{F}$.

Specifically, $\mathcal{G}$ will be defined similarly to $\mathcal{F}$, except that its key will also include a seed s of a pseudo-random function PRF. Instead of mapping the input a to the output b, $g_{\mathsf{k},\mathsf{s}}$ will include two new functions $\mathsf{PRF}_{\mathsf{s}}(x)$ and $\mathbb{G}(x) = b \oplus \mathsf{PRF}_{\mathsf{s}}(x \oplus a)$. Note, that without knowing a, an efficient algorithm that gets black-box access to $\mathsf{PRF}_{\mathsf{s}}$ and $\mathbb{G}$ cannot find any correlation between the functions, and therefore, cannot learn b. However, knowing a, we can query $\mathsf{PRF}_{\mathsf{s}}$ with a random string r, query $\mathbb{G}$ with $r \oplus a$, and obtain $b = \mathsf{PRF}_{\mathsf{s}}(r) \oplus \mathbb{G}(r \oplus a)$. Since each individual query to $\mathcal{G}$ is distributed uniformly, 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 use standard direct product amplification [49]; namely, we let $\mathcal{G}$ evaluate $\mathsf{PRF}_{\mathsf{s}}$ and $\mathbb{G}$ 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 do not output b given an encryption of 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 $b \oplus \mathsf{PRF}_{\mathsf{s}}(x \oplus b)$. However, given black-box access to such a function, semantic security is not maintained. Indeed, when two encryptions c_1 and c_2 correspond to the same plaintext then $\mathbb{G}'(\mathsf{c}_1) = \mathbb{G}'(\mathsf{c}_2)$ and when the encryptions correspond to different plaintexts $\mathbb{G}'(\mathsf{c}_1) \neq \mathbb{G}'(\mathsf{c}_2)$ with high probability. Using such an attack it is possible to learn a and thus also b, in a black-box way.

The first step in our solution involves a slight modification to the homomorphic encryption scheme. An encryption of a bit b will include $b \oplus r$ for a random bit r and an encryption of r. We shall also assume for simplicity that encryptions of uniformly random strings are uniformly random themselves. We also modify the decryption and homomorphic evaluation algorithms accordingly. We refer to such an encryption scheme as *decomposable*, since any cipher c is composed of two parts. We think of the encryption of r as a public part and denote it by $\mathsf{pub}(\mathsf{c})$, and of the padded plaintext as a private part. $\mathsf{pub}(\mathsf{c})$ is independent of the plaintext and can be sampled by anyone, but together with the plaintext, it completely determines the cipher.

The next step is to 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. Additionally, given an encryption of b, it is still possible to learn b by first sampling a random string r and then homomorphically computing an encryption c of $b \oplus r$, and finally, evaluating $\mathbb{G}'(\mathsf{c}) \oplus \mathsf{PRF}_{\mathsf{s}}(r||\mathsf{pub}(\mathsf{c})) = b$. Note that both inputs c and $r||\mathsf{pub}(\mathsf{c})$ to $\mathbb{G}'$ and $\mathsf{PRF}_{\mathsf{s}}$ are distributed uniformly at random.

Getting rid of homomorphic encryption. The construction of Barak et al. only uses a standard symmetric encryption, where homomorphic operations are performed gate by gate by f_{k}: given two encrypted input bits, f_{k} decrypts them, evaluates the desired operation, and returns the result encrypted. Trying to employ this strategy in our setting we may hope that, since these queries consist of random encryptions, a circuit C that does not err too much will correctly compute the homomorphic operations required for extraction with high enough probability. However, this is not the case since extraction involves a long (a-priori unbounded) homomorphic computation, and even one error along the way could compromise the integrity of the result.

Here we use the fact that, in the case of robust unobfuscatable functions, we can assume that C only makes "detectable errors"; we show that this is enough to recover from errors during the homomorphic computation. Specifically, when a single homomorphic operation fails (the circuit outputs $\perp$), we would like 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 $\mathcal{G}$ 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. (In the actual construction, when dealing with a deterministic $\mathcal{G}$, we will need to use "invoker randomizable" PRFs [5] to make sure that the encryptions are indeed random.)

From several (key-dependent) input distributions to the uniform distribution. As described above, a function $g \in \mathcal{G}$ implements several different functions: $\mathsf{PRF}_{\mathsf{s}}, \mathbb{G}, \mathbb{G}'$, and the homomorphic evaluation functionality. For the extraction procedure described above to succeed, it is not sufficient that C approximates g well on a random input, but rather C should approximate g on each one of these functions individually. Moreover, to guarantee the success of the homomorphic evaluation, we require that C approximates the homomorphic evaluation functionality for *every* possible configuration of gate and input plaintexts. Note that for a specific configuration of gate and input plaintexts, the input to homomorphic evaluation functionality is not distributed uniformly, furthermore, such inputs 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.

To overcome this issue, we use an encryption scheme that supports sampling ciphers of random plaintexts independently of the secret key. Intuitively, 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 some related probability. To make sure that the probability is indeed high-enough we use again parallel repetition. Note that during extraction we must query g with encryptions of individual bits; however, we do not sample those encryptions ourself, but rather these encryptions are obtained as the result of a previous homomorphic operation. Eventually, we show that extraction is successful assuming only that C approximates the resulting function on a uniform input.

Necessity of one-way functions. We show that one-way functions are not only sufficient, but also necessary for robust unobfuscatable functions. (The function that maps $(\mathsf{k}, x_1, \ldots, x_m)$ to $(f_{\mathsf{k}}(x_1), \ldots, f_{\mathsf{k}}(x_m), x_1, \ldots, x_m)$ is one-way for $m \gg |\mathsf{k}|$.) This should be contrasted

with non-robust unobfuscatable functions, where a similar implication is not known; in fact, Barak et al. [5] construct (non-robust) unobfuscatable Turing machine (rather than circuit) families without any computational assumptions, whereas robust unobfuscatable functions also imply one-way functions in the Turing machine case. Additionally we do not know what are the minimal assumptions required for verifiable robust unobfuscatable functions.

Constructing verifiable robust unobfuscatable functions. Finally, we show how to "compile" robust unobfuscatable functions into full fledged verifiable robust unobfuscatable functions based on ZAPs [21] and non-interactive commitments. The high level idea is to put a commitment to the key k in the verification key vk and add to the output of the function a ZAP proving that the function was computed correctly using a key that is consistent with vk. 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 [24, 16]. Now, given a program that computes the function correctly on some noticeable fraction of the inputs, we can detect when the output is computed incorrectly and replace it with $\perp$.

1.6. Related Work. The resettably-sound protocol constructed in [9] was based on the unobfuscatable functions of [5] and oblivious transfer. Subsequently, [19] showed how to remove the collision-resistent hashing from the protocols of [4, 20] leading to resettably-sound ZK from one-way functions. The constructions based on robust unobfuscatable function appeared in a subsequent work [10]. The construction of simultaneously-resettable WI arguments from one-way functions [18] was concurrent to [10]. In another concurrent work Goyal [32] presented a new construction of public-coin concurrent ZK, leading also to a new simultaneously-resettable ZK protocol. The subsequent work of [17] constructed a 4-message resettably-sound ZK protocol based on one-way functions.

1.7. Roadmap. In Section 3, we define the notions of robust and error-robust unobfuscatable functions, as well as other weaker notions used in our constructions. In Section 4, we present our main construction of robust unobfuscatable functions. In Section 5, we define verifiable robust unobfuscatable functions and construct them based on robust unobfuscatable functions. The construction of verifiable unobfuscatable functions will be used to get error-robust unobfuscatable functions and applications to ZK. In Section 6, we describe our new constructions of resettably-secure protocols. In Section 7, we give a transformation from any resettably-sound ZK argument to a family of unobfuscatable functions.

2. Preliminaries.

2.1. Standard Computational Concepts. We rely on the standard notions of Turing machines and Boolean circuits. We say that a (uniform) Turing machine is PPT if is probabilistic and runs in polynomial time. A polynomial-size (or just polysize) circuit family $\mathcal{C}$ is a sequence of circuit $\mathcal{C} = \{C_n\}_{n\in\mathbb{N}}$, such that each circuit C_n is of polynomial size $n^{O(1)}$ and has $n^{O(1)}$ inputs and outputs bits. We follow the standard habit of modeling any efficient adversary strategy as a family of polynomial-sized circuits. For an adversary $\mathcal{A}$ corresponding to a family of polysize circuits $\{\mathcal{A}_n\}_{n\in\mathbb{N}}$, we often omit the subscript n, when it is clear from the context.

A function $\nu : \mathbb{N} \to [0,1]$ is said to be negligible if $\nu(n) = n^{-\omega(1)}$, i.e. ν decays faster than any polynomial. We say that an event occurs with overwhelming probability if it occurs with probability $1 - \nu(n)$ for some negligible function ν. We often denote by negl an unspecified negligible function (e.g., when we say that for all $n \in \mathbb{N}$ and event A_n occurs with probability at most $\mathrm{negl}(n)$, we mean that this is the case for some negligible function). We say that a function $\varepsilon : \mathbb{N} \to [0,1]$ is noticeable if $\varepsilon(n) = n^{-O(1)}$. We denote by $\mathrm{poly}(n)$ an unspecified polynomial.

2.2. A Useful Probability Bound. We prove a basic probability bound that will be used more than once throughout the paper.

The bound. Let $S_1, \ldots, S_m$ be random variables (intuitively, describing an m-step process), and let $G_1, \ldots, G_m$ be events where G_i is determined by S_i alone. Denote by H_i the event that the first $i-1$ stages $S_1, \ldots, S_{i-1}$ are such that that the event G_i occurs with bounded probability δ (when S_i is sampled conditioned on $S_1, \ldots, S_{i-1}$); namely, H_i is the event that $S_1, \ldots, S_{i-1}$ satisfy:

$$\Pr_{S_i | S_1, \ldots, S_{i-1}} [G_i \mid S_1, \ldots, S_{i-1}] \leq \delta \ .^2$$

The following intuitive claim shows an exponential decay in the probability that k events G_i occur when the residual probability of their occurrence is bounded through every stage of the process.

CLAIM 2.1. *For every* $\{i_1, i_2, \ldots, i_k\} \subseteq [m]$*:*

$$\Pr_{S_1, \ldots, S_m} \left[\left(\bigwedge_{j=1}^{k} G_{i_j} \right) \wedge \left(\bigwedge_{j=1}^{m} H_j \right) \right] \leq \delta^k \ .$$

Proof. First, we show by induction that for any $0 \leq \ell \leq k$:

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

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

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

[2]More explicitly, we mean that H_i is that even that $(S_1, \ldots, S_{i-1}) = (s_1, \ldots, s_{i-1})$, such that $(s_1, \ldots, s_{i-1})$ satisfy $\Pr_{S_i | (S_1, \ldots, S_{i-1}) = (s_1, \ldots, s_{i-1})} [G_i \mid (S_1, \ldots, S_{i-1}) = (s_1, \ldots, s_{i-1})] \leq \delta$.

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

$$\Pr\left[\left(\bigwedge_{j=1}^{k} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{m} H_j\right)\right] \leq \Pr\left[\left(\bigwedge_{j=1}^{k} G_{i_j}\right) \wedge \left(\bigwedge_{j=1}^{k} H_{i_j}\right)\right] \leq \delta^k \ .$$

□

3. Robust Unobfuscatable Functions. In this section, we define the notion of robust unobfuscatable functions. Before that we define the main notion of function approximation that we work with.

DEFINITION 3.1 ($(\varepsilon, \nu, \mathcal{D})$-approximation). *Let $\mathcal{D}$ be a distribution on $\{0,1\}^n$. A circuit C is said to $(\varepsilon, \nu, \mathcal{D})$-approximate a function $f : \{0,1\}^n \to \{0,1\}^{n'}$ if:*

1. **It often agrees with f on inputs drawn from $\mathcal{D}$:**

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

2. **It almost only makes detectable errors on inputs from $\mathcal{D}$:**

$$\Pr_{q \leftarrow \mathcal{D}}[C(q) \notin \{\perp, f(q)\}] \leq \nu \ .$$

C is said to $(\varepsilon, \mathcal{D})$-approximate f **with errors** *if property 2 is not guaranteed.*

DEFINITION 3.2 (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 an NP relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$, if it is:*

1. **Black-box unlearnable:** *For any poly-size learner L, and any $n \in \mathbb{N}$, L:*

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

   namely, L cannot satisfy the relation $\mathcal{R}_{\mathcal{F}}$, given only black-box access to f_{k}. $\mathcal{R}_{\mathcal{F}}$ is thus called "the unlearnable relation".
2. **Non-Black-box learnable:** *There exists an efficient extractor E such that, for any functions $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$ and $\nu = \nu(n) \leq \frac{\varepsilon}{4n^2}$, any efficient sampler $\mathcal{D} = \mathcal{D}(1^n)$, any large enough $n \in \mathbb{N}$, $\mathsf{k} \in \{0,1\}^n$, and circuit C that $(\varepsilon, \nu, \mathcal{D})$-approximates f_{k}, E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from C:*

$$\Pr_{\mathsf{E}}\left[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n, 1^{1/\varepsilon})\right] \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)}\right) \cdot \left(|C| \cdot n/\varepsilon\right)^{O(1)} \ .^3$$

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

$$\Pr_{\mathsf{E}}[(\mathsf{k}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n)] \geq 1 - \left(\nu^{\Omega(1)} + 2^{-\Omega(n)}\right) \cdot \left(|C| \cdot n\right)^{O(1)} \ .$$

[3] In most cases, the extraction error expression above can be simplified to $\mathrm{negl}(n)$, e.g., when $|C| = \mathrm{poly}(n)$, $\varepsilon = 1/\mathrm{poly}(n)$, and $\nu = \mathrm{negl}(n)$. There are however cases in which we may use the more general bound above, e.g., in Section 6.3.2.

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 3.3 (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 an NP relation $\mathcal{R}_{\mathcal{F}}$ and input sampler $\mathcal{D}$, if it is:*

1. **Black-box unlearnable:** *As in Definition 3.2.*
2. **Non-Black-box learnable:** *There exists an efficient extractor E such that, for any function $\varepsilon = \varepsilon(n) \geq 2^{-\sqrt{n}}$, any efficient sampler $\mathcal{D} = \mathcal{D}(1^n)$, all large enough $n \in \mathbb{N}$, and any $\mathsf{k} \in \{0,1\}^n$, E extracts $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from any circuit C that $(\frac{1}{2}+\varepsilon, \mathcal{D})$-approximates f_{k}* **with errors***:*

$$\Pr_{\mathsf{E}}\left[(\mathsf{k},\mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C, 1^n, 1^{1/\varepsilon})\right] \geq 1 - 2^{-\Omega(n)} \cdot |C|^{O(1)} \ .$$

REMARK 3.1 (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 strings in $\{0,1\}^n$. Indeed, our constructions also achieve this variant (see Remark 4.1).*

REMARK 3.2 (Uniquely determined functions). *For the case of erroneous circuits, i.e. with non-detectable errors (Definition 3.3), we require for simplicity that the circuit C determines one specific function, and hence we require more than $\frac{1}{2}$-agreement (this is necessary to guarantee that C cannot simultaneously approximate two functions in the family, and sufficient assuming that any two functions in the family have no 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 3.3 (unique-witness unlearnable relations). *A natural requirement regarding the unlearnable relation $\mathcal{R}_{\mathcal{F}}$ of a family $\mathcal{F}$ is that there exists a unique witness $\mathsf{z} \in \mathcal{R}_{\mathcal{F}}(\mathsf{k})$; namely, z is a deterministic function of k. Furthermore, by the extraction guarantee, we can further assume that, given k in the clear, the corresponding witness z can be computed in expected polynomial time. Our constructions satisfy the unique-witness requirement, and also allow to compute the unique witness in strict polynomial time.*

We also note that this property allows to rule out the same notion of virtual black-box obfuscation ruled out in [5] that prevents the adversary from learning any deterministic function of the key k.

REMARK 3.4 (Extraction from computationally-bounded circuits). *As stated above extraction in Definitions 3.2 and 3.3 is information theoretic, and does not explicitly require that the approximating circuit C is of poly-size (although this is the typical case in applications).*

Here there is a difference between Definition 3.2 and Definition 3.3. In Definition 3.2, the fraction ν of undetectable errors is explicit in the definition and affects the probability that extraction will indeed succeed; in particular, we do not require that ν is a concrete negligible function. Jumping ahead, when extracting from poly-size circuits, we may prevent undetectable errors using, for example, a computationally-sound proof system, in which case the function ν may be some negligible function $\mathrm{negl}(n)$ that depends on the concrete poly-size family of circuits we are extracting from. In Definition 3.3, we require that extraction

works even with undetectable errors, and as said we do not a-priori restrict the circuits to poly-size. To achieve such extraction, we will indeed rely on statistically-sound proofs systems (concretely ZAPs). It is possible to augment the definition to only consider poly-size circuit families, in which case there will be some negligible error $\mathrm{negl}(n)$ *in the extraction probability, which depends on the concrete family of circuits.*

REMARK 3.5 (Comparison with strongly unobfuscatable functions of [5]). *Barak et al. define and construct a family of functions that is strongly unobfuscatable. Such functions satisfy a weaker extraction requirement than error-robust unobfuscatable functions, and therefore only rule out a stronger definition of approximate obfuscation.*

The definition of [5] considers extraction from a circuit that is sampled from a distribution $\mathcal{C}$ *over circuits (this corresponds to the output distribution of an obfuscator on a specific input program with random coins). The extraction requirement states that if, for every input* q *a circuit* $C \leftarrow \mathcal{C}$ *satisfies* $C(q) = f_{\mathsf{k}}(q)$ *with overwhelming probability over the choice of* C*, then the extractor on input circuit* $C \leftarrow \mathcal{C}$ *outputs* k *with overwhelming probability. In particular, extraction requires that, for* every *input, the circuit is correct with overwhelming probability. In contrast, the extraction requirement of error-robust unobfuscatable functions is guaranteed even if for some inputs the circuit never answers correctly (in fact this could be the case for almost all inputs).*

3.1. 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 3.2. We restrict attention to approximation **without errors**. (Eventually, in Section 5, we show how to go from robust functions to error-robust ones.)

First, we show that, using standard direct product amplification [49], we can always boost the robustness of an unobfuscatable family (at the cost of blowing up the input and output size).

LEMMA 3.4. *Any* $\left(1 - \frac{1}{\sqrt[4]{n}}\right)$*-robust* $\mathcal{G}$ *can be transformed to a robust* $\mathcal{F}$*, where the input and output size grows by a factor of* n*. Furthermore, if the extractor of* $\mathcal{G}$ *runs in time* $\mathrm{poly}(n) \cdot |C|^d$*, where* C *is the approximation circuit for* g_{k}*, then the extractor of* $\mathcal{F}$ *runs in time* $\mathrm{poly}(n) \cdot (|C|/\varepsilon)^d$*, where* C *is a* $(\varepsilon, \nu, \mathcal{D})$*-approximation circuit for* f_{k}*.*

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

CONSTRUCTION 3.1 (Amplifying 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* $\widehat{\mathcal{D}} = \mathcal{D} \times \ldots \times \mathcal{D}$*. 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* $g_{\mathsf{k}} : \{0,1\}^n \to \{0,1\}^n$*.*

Proof of Lemma 3.4. Unlearnability of $\mathcal{F}$ follows directly from that of $\mathcal{G}$, since any oracle call to $f_{\mathsf{k}} \in \mathcal{F}$ 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 functions $\varepsilon(n) \geq 2^{-\sqrt{n}}$, $\nu(n) \leq \frac{\varepsilon}{4n^2}$ the extractor $\widehat{\mathsf{E}}$ is given as input a circuit C that $\left(\varepsilon, \nu, \widehat{\mathcal{D}}\right)$-approximates f_{k}, as well as the parameter $1^{1/\varepsilon}$. $\widehat{\mathsf{E}}$ first constructs from C a probabilistic circuit C' that, with very high probability, will give a very good approximation of g_{k}.

The circuit C', given a random sample $q \leftarrow \mathcal{D}(1^n)$, samples $(q_1, \ldots, q_n) \leftarrow \widehat{\mathcal{D}}(1^n)$, replaces a random coordinate q_i with q, and feeds the augmented n-fold sample to C. The circuit C' repeats this procedure $\frac{n^2}{\varepsilon}$ times (for the same q), and if it obtains an answer

$(a_1, \ldots, a_n)$ that is not $\perp$, it returns a_i. Then, after producing the circuit C', $\widehat{\mathsf{E}}$ chooses n random strings $r_1, \ldots, r_n$ for C', and runs the extractor E of g_{k}, on each $C'(\cdot; r_i)$. 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' is likely to approximate g_{k} sufficiently well. 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 $\Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}|_q}[C(\vec{q}) \neq \perp] \geq \varepsilon/2n$.

We claim that a sample q drawn from $\mathcal{D}$ is good with probability at least $1 - \frac{\log(2/\varepsilon)}{n} \geq 1 - O\left(\frac{1}{\sqrt{n}}\right)$. Indeed, if that was not the case, then in a sample $\vec{q} \leftarrow \widehat{\mathcal{D}}(1^n)$, there is some coordinate that is **not** good, except with probability $\varepsilon/2$; more precisely,

$$\Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}}[q_1, \ldots, q_n \in \mathsf{good}] \leq \prod_{i \in [n]} \Pr_{q_i \leftarrow \mathcal{D}}[q_i \in \mathsf{good}] \leq \left(1 - \frac{\log(2/\varepsilon)}{n}\right)^n \leq \varepsilon/2 \ .$$

However, the probability that $C(\vec{q}) \neq \perp$ and some q_i is not good is at most $\varepsilon/2$; indeed,

$$\begin{aligned}
&\frac{1}{n} \cdot \Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}}[C(\vec{q}) \neq \perp, \exists i : q_i \notin \mathsf{good}] \leq \\
&\quad \Pr_{\vec{q} \leftarrow \widehat{\mathcal{D}}, i \leftarrow [n]}[C(\vec{q}) \neq \perp, q_i \notin \mathsf{good}] \leq \\
&\quad \Pr_{q \leftarrow \mathcal{D}, \vec{q} \leftarrow \widehat{\mathcal{D}}|_q}[C(\vec{q}) \neq \perp, q \notin \mathsf{good}] \leq \\
&\quad \Pr_{q \leftarrow \mathcal{D}|\overline{\mathsf{good}}, \vec{q} \leftarrow \widehat{\mathcal{D}}|_q}[C(\vec{q}) \neq \perp] \leq \\
&\quad \varepsilon/2n \ ,
\end{aligned}$$

where $\mathcal{D}|\overline{\mathsf{good}}$ denotes $\mathcal{D}$ conditioned on not hitting the set good. Overall $C(\vec{q}) \neq \perp$ with probability less than $\varepsilon/2 + \varepsilon/2 \leq \varepsilon$, leading to a contradiction.

Next, note that, conditioned on q being good, $C'(q) \neq \perp$, except with probability $\left(1 - \frac{\varepsilon}{2n}\right)^{n^2/\varepsilon} \leq 2^{-n/2}$, and thus

$$\Pr_{q \leftarrow \mathcal{D}, r}[C'(q; r) \neq \perp] \geq \left(1 - O\left(\frac{1}{\sqrt{n}}\right)\right) \cdot \left(1 - 2^{-\Omega(n)}\right) \geq \left(1 - O\left(\frac{1}{\sqrt{n}}\right)\right) \ .$$

In addition, since $C(\vec{q}) \notin \{\perp, \widehat{g}_{\mathsf{k}}(\vec{q})\}$ with probability at most $\nu(n)$, by a union bound

$$\Pr_{q \leftarrow \mathcal{D}, r}[C'(q; r) \notin \{\perp, g_{\mathsf{k}}(q)\}] \leq \nu(n) \cdot \frac{n^2}{\varepsilon} \leq \frac{1}{4} \ .$$

Combining these two facts, it follows that for a $\left(1 - \frac{1}{\sqrt[4]{n}} - \frac{1}{2}\right)$-fraction of the random coins r used by C', it holds that both

$$\begin{aligned}
\Pr_{q \leftarrow \mathcal{D}}[C'(q; r) \neq \perp] \quad &\geq 1 - O\left(\frac{1}{\sqrt[4]{n}}\right) \\
\Pr_{q \leftarrow \mathcal{D}}[C'(q; r) \notin \{\perp, \widehat{g}_{\mathsf{k}}(\vec{q})\}] &\geq 1 - \sqrt{\nu(n) \cdot \frac{n^2}{\varepsilon}} \ ;
\end{aligned}$$

namely, $\mathcal{C}'(\cdot;r)$ is a $\left(1-O\left(\frac{1}{\sqrt[4]{n}}\right), \sqrt{\nu(n)\cdot\frac{n^2}{\varepsilon}}, \mathcal{D}\right)$-approximation of g_{k}. For such randomness r, we know that the extractor $\mathsf{E}_{\mathcal{G}}$ for g_{k} outputs k with probability at least

$$1-\left(\sqrt{\nu(n)\cdot\frac{n}{\varepsilon}}\right)^{\Omega(1)}\cdot(|C'|\cdot n)^{O(1)} = 1-\nu^{\Omega(1)}\left(|C|\cdot n/\varepsilon\right)^{O(1)} \quad .$$

and since there are n trials, the overall extraction probability is

$$1-\nu^{\Omega(1)}\left(|C|\cdot n/\varepsilon\right)^{O(1)} - \left(1-\frac{1}{2}-\frac{1}{\sqrt[4]{n}}\right)^{n} \geq 1-\left(\nu^{\Omega(1)}+2^{-\Omega(n)}\right)\cdot\left(|C|\cdot n/\varepsilon\right)^{O(1)} \quad ,$$

as required.

Noting that the size of C' is $\mathrm{poly}(n)\cdot|C|/\varepsilon$, and that we make n trials to extract from C'; we deduce that, if $\mathsf{E}_{\mathcal{G}}$ runs in time $\mathrm{poly}(n)\cdot|C|^d$, then the extractor $\mathsf{E}_{\mathcal{F}}$ runs in time $\mathrm{poly}(n)\cdot(|C|/\varepsilon)^d$. □

When constructing robust unobfuscatable functions, it will be convenient to also consider a further relaxed notion where, in order to extract, the circuit is required to approximate the function on multiple distributions $\mathcal{D}_1,\ldots,\mathcal{D}_d$, instead of one. Throughout, 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 3.2, where $d=1$. As another relaxation, we will not require that each of these distributions 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 later on, 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 some concrete constant approximation factor $1-\delta$ (we will eventually set $\delta < 2^{-5}$, see Section 4).

DEFINITION 3.5 (d-distribution robust unobfuscatable function). *A family of functions $\mathcal{F}$ is a d-distribution robust unobfuscatable family with respect to an NP 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 3.2.*
2. **Non-Black-box learnable:** *There exists an efficient extractor E such that, for any function $\nu=\nu(n)$, for any large enough $n\in\mathbb{N}$, any $\mathsf{k}\in\{0,1\}^n$, and every circuit C that $(1-\delta,\nu,\mathcal{D}_i)$-approximates f_{k}, for all $i\in[d]$, E extracts $\mathsf{z}\in\mathcal{R}_{\mathcal{F}}(\mathsf{k})$ from C:*

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

where $\mathcal{D}_i=\mathcal{D}_i(1^n,\mathsf{k})$.

DEFINITION 3.6 (jointly key-independent distributions). *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 $\mathcal{D}_1,\ldots,\mathcal{D}_d$ are **jointly key-independent***, if there exists an efficient sampler $\mathcal{D}^*$ such that for all k, $\mathcal{D}^*(1^n)$ generates samples from the distribution $\mathcal{D}^*_{\mathsf{k}}$.*

We show that any d-distribution robust unobfuscatable family, where $\mathcal{D}_1,\ldots,\mathcal{D}_d$ are jointly key-independent, can be transformed into a robust unobfuscatable family according

to Definition 3.2. Specifically, we show a $\left(1-\frac{\delta}{d}\right)$-robust family, and then robustness can be amplified using Lemma 3.4.

LEMMA 3.7. *If $\mathcal{G}$ is a d-distribution robust unobfuscatable family, with samplers $\mathcal{D}_1,\ldots,\mathcal{D}_d$ that are jointly key-independent, then $\mathcal{G}$ is also $\left(1-\frac{\delta}{d}\right)$-robust with respect to the (key-independent) distribution $\mathcal{D}^*=\mathcal{D}^*(1^n)$.*

Proof. Indeed, by a simple union bound, any circuit that $\left(1-\frac{\delta}{d},\nu,\mathcal{D}^*\right)$-approximates f_{k} must $(1-\delta,\nu,\mathcal{D}_i)$-approximate f_{k} for each one of the distributions $\mathcal{D}_1,\ldots,\mathcal{D}_d$. Thus, we can use the existing extractor associated with $\mathcal{G}$. □

3.2. Robust Unobfuscatable Functions with a Hardcore Secret. The notion of robust (and error-robust) unobfuscatable functions 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 5), and for some applications to resettable protocols (see Section 6.5).

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 (called hardcore functions).

DEFINITION 3.8 (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 L, and any $n\in\mathbb{N}$:*

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

   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 functions $\varepsilon=\varepsilon(n)\geq 2^{-\sqrt{n}}$, $\nu=\nu(n)\leq\frac{\varepsilon}{4n^2}$, any efficient sampler $\mathcal{D}=\mathcal{D}(1^n)$, any large enough n, any hardcore function $h\in\mathcal{HC}_n$, any $\mathsf{k}\in\{0,1\}^n$, and every circuit C that $(\varepsilon,\nu,\mathcal{D})$-approximates f_{k}, E extracts $h(\mathsf{k})$ from C:*

$$\Pr_{\mathsf{E}}\left[h(\mathsf{k})\leftarrow\mathsf{E}(C,h,1^n,1^{1/\varepsilon})\right]\geq 1-\left(\nu^{\Omega(1)}+2^{-\Omega(n)}\right)\cdot\left(|C|\cdot n/\varepsilon\right)^{O(1)}\ .$$

We show that any robust unobfuscatable family $\mathcal{G}$ that has a unique-witness unlearnable relation $\mathcal{R}_{\mathcal{G}}$ (as defined in Remark 3.3) can be transformed into a robust $\mathcal{F}$ with a hardcore secret.

LEMMA 3.9. *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 relation $\mathcal{R}_{\mathcal{F}}$ and the hardcore family $\mathcal{HC}$.*

Proof. 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 $\widehat{\mathcal{D}}=\mathcal{D}\times\ldots\times\mathcal{D}$, and $f_{\mathsf{k}}(q_1,\ldots,q_n)$ 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 [29] 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 3.8), note that the secrets $\mathsf{z}_1, \ldots, \mathsf{z}_n$ are each unlearnable 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 [29] 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 3.8; namely, the hardcore secret can be extracted from any circuit approximation. Indeed, note that any circuit C that $(\varepsilon, \nu, \widehat{\mathcal{D}})$-approximates f_{k} can be transformed, with overwhelming probability, into a circuit that $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$-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 $\varepsilon/2 - \sqrt{\nu} \geq \varepsilon/4$ over the choice of randomness r for C' the resulting circuit $C'(\cdot; r)$ $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$-approximates $g_{\mathsf{k}'_i}$. Thus we can take n/ε random copies of C', and with probability $1 - 2^{-n}$ ensure that one of the circuits $(\varepsilon/2, \sqrt{\nu}, \mathcal{D})$-approximates $g_{\mathsf{k}'_i}$.

Thus the extractor $\mathsf{E}_{\mathcal{F}}$ for $\mathcal{F}$, would run the extractor $\mathsf{E}_{\mathcal{G}}$ with each of the n instances of circuit C', and obtain the corresponding secret $\mathsf{z}_i \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}'_i)$, with probability $1 - \nu^{\Omega(1)}\left(|C| \cdot n/\varepsilon\right)^{O(1)}$. 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 3.9. □

4. A Construction of Robust Unobfuscatable Functions. In this section, we construct robust unobfuscatable functions from one-way functions.

THEOREM 4.1. *Assuming one-way functions, there exists a family of robust unobfuscatable functions.*

4.1. Required PRFs and Encryption. Before describing the construction, we define several required primitives.

Invoker randomizable pseudo-random functions. Invoker randomizable pseudo-random functions 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 [5].

DEFINITION 4.2 (Invoker randomizable pseudo-random functions [5]). *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.* [5] show that invoker randomizable PRFs are implied by any PRF.

Decomposable CCA-1 symmetric-key encryption. In our construction, we will make use of *symmetric-key encryption* schemes with specific properties. We next define these properties and note existing constructions satisfying them, 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 random plaintexts. Such an encryption scheme can be obtained using a one-bit output PRF by encrypting $\mathsf{Enc}_{\mathsf{sk}}(b;r) = (r, b \oplus \mathsf{PRF}_{\mathsf{sk}}(r))$. 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 ciphertext 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 4.3 (Decomposable encryption). *An encryption scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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* PubSamp *such that, for any secret key* $\mathsf{sk} \in \{0,1\}^n$*:*

$$\mathsf{PubSamp}(1^n) = \mathsf{pub}(\mathsf{Enc}_{\mathsf{sk}}(0)) = \mathsf{pub}(\mathsf{Enc}_{\mathsf{sk}}(1)) \ .$$

- *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 4.4 (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* RanSamp *such that for any secret key* $\mathsf{sk} \in \{0,1\}^n$*:*

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

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

We remark that in both Definitions 4.3 and 4.4, 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 PubSamp, can be simply implemented as $\mathsf{PubSamp}(1^n) = \mathsf{pub}(\mathsf{RanSamp}(1^n))$, 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 [26]), it is clearly decomposable and has random cipher generation.

4.2. The Construction. To prove Theorem 4.1, we construct an unobfuscatable function family that is d-distribution robust with respect to distribution ensembles $(\mathcal{D}_1, \ldots, \mathcal{D}_d)$ that are jointly key-independent in the sense of Definition 3.6. Then, by Lemmas 3.7 and 3.4 we can obtain robust unobfuscatable functions according to Definition 3.2.

Our construction follows the high-level overview provided in Section 1.5, we now proceed to describe it in detail.

CONSTRUCTION 4.1. *Let (a, b) be strings, sk a key for a CCA-1 symmetric key for a decomposable encryption scheme with random ciphertext generation (Definitions 4.3,4.4), and let s a be seed for a pseudo-random function PRF. The construction will use two probabilistic and three deterministic auxiliary functions:*

1. *The function $\mathbb{A}_{\mathsf{sk},a}$, given any 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 a plain text $r \in \{0,1\}^n$, and returns $b \oplus \mathsf{PRF}_{\mathsf{s}}(r \oplus b || \mathsf{pub}(\mathsf{c}_1) || \ldots || \mathsf{pub}(\mathsf{c}_n))$, where $\mathsf{pub}(\mathsf{c})$ is the public part of c.*

Making the functions deterministic and invoker randomizable. Let s' be a seed for an invoker randomizable PRF. 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 $r \in \{0,1\}^{\mathrm{poly}(n)}$. $\mathbb{G}_{\mathsf{key},\mathsf{s}'}(q; r)$ runs the original $\mathbb{G}_{\mathsf{key}}(q)$, using randomness $\mathsf{PRF}_{\mathsf{s}'}(q; r)$.

The n-fold functions $\widehat{\mathbb{A}}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}$. We define variants of $\mathbb{A}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}_{\mathsf{sk},\mathsf{s}'}$, that instead of returning a single cipher, return an n-fold cipher, consisting of n independent ciphers encrypting the same result. Specifically, for $\mathbb{G} \in \{\mathbb{A}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}_{\mathsf{sk},\mathsf{s}'}\}$, $\widehat{\mathbb{G}}$ gets input $(q, r_1, \ldots, r_n)$ and returns $\mathbb{G}(q, r_1), \ldots, \mathbb{G}(q, r_n)$.

The actual 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 $\mathbb{G} \in \left\{\widehat{\mathbb{A}}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}\right\}$ 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 in the sense of Remark 3.3.)

4.3. Black-Box Unlearnability. We now prove black-box unlearnability. The high-level overview of the proof is presented in Section 1.5.

LEMMA 4.5. *$\mathcal{F}$ given by Construction 4.1 satisfies black-box unlearnability with respect to $\mathcal{R}_{\mathcal{F}}$.*

Proof. We shall perform the analysis in several steps: we will first show unlearnability assuming probabilistic oracles, where $\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}$ are given in their 1-fold version; then, we

will show security for the case that randomness is derived using PRFs; finally, we will move to security for n-fold $\widehat{\mathbb{A}}_{\mathsf{sk},a}, \widehat{\mathbb{H}}_{\mathsf{sk}}$, and for the actual function f_{k}.

Let $\mathcal{A}$ be a polysize adversary with oracle access to the functions $\mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$. Let E_1 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

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

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

CLAIM 4.1. $\Pr_{b,\mathsf{sk},\mathsf{s}}[E_1] = 0$.

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 $q = q'$; indeed, recall that, in decomposable encryption (Definition 4.3), the public part and the plaintext determine the cipher. □

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

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 $\neg E^i$ be the event that E_2 does **not** occur in the first i queries, it is enough to show that

$$\min_{i \leq |\mathcal{A}|} \Pr\left[\neg E^{i+1} | \neg E^i\right] \geq 1 - \mathrm{negl}(n) \ .$$

Indeed, conditioned on $\neg E^i$, the view of $\mathcal{A}$ after the first i queries is information theoretically independent of b. So to show that $\neg E^{i+1}$ occurs with overwhelming probability, it is enough to show that, if it does not occur, $\mathcal{A}$ can reconstruct b from its view in the first i queries. 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, even given the oracles $\mathbb{H}_{\mathsf{sk}}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$, the encryption scheme using sk is still semantically secure.

CLAIM 4.3 (semantic security). *Let $\mathcal{A}$ be a polysize distinguisher, then:*

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

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

Proof. We first note that by Claims 4.1,4.2, we can replace $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ (in both distributions) with a random function that is completely independent of sk. Now, the claim follows directly from the CCA-1 security of the encryption scheme, just as in [5, Claim 3.7]. □

Next, 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 4.4. $\Pr_{b,\mathsf{sk},\mathsf{s}}[E] \leq \mathrm{negl}(n)$.

Proof. First, we note that the event E is testable given a and thus, it is enough to show that the claim holds when PRF_s is replaced by a truly random function R. Now, assume towards contradiction that E occurs with noticeable probability $\varepsilon = \varepsilon(n)$. 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 $\varepsilon/|\mathcal{A}|$. 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 $\mathcal{A}$ still has the same distribution because the first i answers to $\mathbb{R}_{a,b,R}$ are uncorrelated to the answers of $\mathbb{R}_R$. The latter is true since the queries are not correlated by a, and they're also uncorrelated with the answers of $\mathbb{R}_{\mathsf{sk},b,R}$, which can be thought of as an independent function, because it has different input length.

Thus, in this hybrid experiment, E still occurs for the first time in the i'th query with probability $\varepsilon/|\mathcal{A}|$. Now, after replacing $\mathbb{R}_{a,b,R}$ with an independent random function R', for the first $i-1$ queries, we can think about a new adversary $\mathcal{A}_i$ that halts after making the i'th query. By semantic security Claim 4.3 (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 affecting the probability that E occurs 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 two 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 PRF_s with a truly random function R; then, by Claim 4.4, we can replace $\mathbb{R}_{a,b,R}$ with an independent random function R'; now, we can invoke semantic security (Claim 4.3) to replace $\mathbb{A}_{\mathsf{sk},a}$ with $\mathbb{A}_{\mathsf{sk},0^n}$, and finally we can invoke again Claims 4.1,4.2, to replace $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ with another independent random function R''. Overall, each change affects the probability that $\mathcal{A}$ outputs b, only by a negligible amount; 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, in the function f_k, $\mathbb{A}_{\mathsf{sk},a}, \mathbb{H}_{\mathsf{sk}}$ are replaced with invoker randomizable deterministic n-fold variants. 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 $\mathbb{G}$.

This completes the proof of Lemma 4.5. □

4.4. Non-Black-Box Learnability. We now prove that Construction 4.1 is non-black-box learnable in the sense of Definition 3.5. The high-level ideas behind the proof are given in Section 1.5.

We describe the extraction procedure. As a first step, we describe an extractor that is only required to work for circuits that perfectly compute each one of the underlying functions, and where $\mathbb{A}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}_{\mathsf{sk},\mathsf{s}'}$ are given in their 1-fold version. Then, we will explain how to generalize this extractor for the case of faulty circuits, where $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ are in their n-fold version.

CONSTRUCTION 4.2. *Let C be a circuit that perfectly implements a function $f_\mathsf{k} \in \mathcal{F}$, consisting of circuits $\left\{\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}, \mathbb{R}^{\triangle}_{\mathsf{s}}, \mathbb{R}^{\triangle}_{a,b,\mathsf{s}}, \mathbb{R}^{\triangle}_{\mathsf{sk},b,\mathsf{s}}\right\}$ that perfectly implement each of*

f_{k}'s underlying functions. Let $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(U)$ and $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot;U)$ denote the distributions of $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(r)$ and $\mathbb{H}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot,r)$, when invoked with uniform invoker randomness r.

The extractor E works according to the following steps:

Step 1: *obtain an encryption c_a of a from $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(U)$.*

Step 2: *construct a circuit $C_{a,b}$ that maps a to b, and homomorphically evaluate it on c_a, using $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\cdot;U)$, to obtain an encryption c_b of b. The circuit $C_{a,b}$ is constructed according to Algorithm (1).*

Algorithm 1 The a to b circuit - $C_{a,b}$

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

1: sample $r \leftarrow \{0,1\}^n$
2: obtain $\mathsf{ans} \leftarrow \mathbb{R}^{\triangle}_{\mathsf{s}}(r)$
3: obtain $\mathsf{ans}' \leftarrow \mathbb{R}^{\triangle}_{a,b,\mathsf{s}}(\tilde{a}\oplus r)$
4: **return** $\mathsf{ans}\oplus\mathsf{ans}'$

By the definition of the oracles $\mathbb{R}^{\triangle}_{a,b,\mathsf{s}}(\tilde{a}\oplus r)\oplus\mathbb{R}^{\triangle}_{\mathsf{s}}(r) = b\oplus\mathsf{PRF}_{\mathsf{s}}(a\oplus\tilde{a}\oplus r)\oplus\mathsf{PRF}_{\mathsf{s}}(r)$, which is indeed b when $\tilde{a}=a$.

Step 3: *transform the encryption c_b of b into b according to the procedure C_b, given by Algorithm (2).*

Algorithm 2 The b decoding procedure - C_b

Input: An encryption c of $\tilde{b}$ (allegedly $\tilde{b}=b$)

1: sample $r \leftarrow \{0,1\}^n$
2: using $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\cdot;U)$, homomorphically compute from c a new encryption c' of $\tilde{b}\oplus r$
3: obtain $\mathsf{ans} \leftarrow \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}(\mathsf{c}')$
4: obtain $\mathsf{ans}' \leftarrow \mathbb{R}_{\mathsf{s}}(r||\mathsf{pub}(\mathsf{c}'))$
5: **return** $\mathsf{ans}\oplus\mathsf{ans}'$

By the definition of the oracles $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}(\mathsf{c}')\oplus\mathbb{R}_{\mathsf{s}}(r||\mathsf{pub}(\mathsf{c}')) = b\oplus\mathsf{PRF}_{\mathsf{s}}(b\oplus\tilde{b}\oplus r||\mathsf{pub}(\mathsf{c}'))\oplus\mathsf{PRF}_{\mathsf{s}}(r||\mathsf{pub}(\mathsf{c}'))$, which is indeed b when $\tilde{b}=b$.

Extraction from circuits with detectable errors. We now move to describe our actual extractor $\widehat{\mathsf{E}}$ that can extract b, even from a circuit C that approximates the underlying functions related to $f_{\mathsf{k}}\in\mathcal{F}$, and where $\mathbb{A}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ are given in their n-fold versions. The distributions on which the approximation holds are described following the construction, and are not needed for the description of the extractor, but only for its analysis.

CONSTRUCTION 4.3. *Let C be a circuit that perfectly implements a function $f_{\mathsf{k}}\in\mathcal{F}$, consisting of circuits $\left\{\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}, \mathbb{R}^{\triangle}_{\mathsf{s}}, \mathbb{R}^{\triangle}_{a,b,\mathsf{s}}, \mathbb{R}^{\triangle}_{\mathsf{sk},b,\mathsf{s}}\right\}$ that perfectly implement each of f_{k}'s underlying functions. Let $\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(U)$ and $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot;U)$ denote the distributions of $\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(r_1,\ldots,r_n)$ and $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}(\mathsf{c},\mathsf{c}',\odot,r_1,\ldots,r_n)$, when invoked with uniform invoker randomness $r_1,\ldots,r_n$.*

The extractor $\widehat{\mathsf{E}}$ runs a robust "parallel emulation" of E. The emulation follows the same steps as the extraction above only that it augments each one of them. We first describe how homomorphic ciphertext operations are emulated, and then describe how to augment each of the extraction steps.

Homomorphic operations on n-fold ciphertexts. In the parallel emulation, E is given access to the n-fold versions $\widehat{\mathbb{A}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}$ and $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},a,\mathsf{s}'}$ of the cipher producing oracles. These functions output n-fold ciphertexts; namely, a ciphertext for any single bit m is, in fact, an n-fold cipher consisting of n (1-fold) ciphers, each encrypting the bit m. We shall think of n-fold ciphers encrypting n-bit strings as a matrices, where each row is an n-fold encryption of a bit, and each column is a 1-fold encryption of the entire string.

In the parallel emulation, homomorphic operations are done on n-fold bit encryptions. Specifically, given two n-fold ciphers $\vec{\mathsf{c}} = (\mathsf{c}_1, \ldots, \mathsf{c}_n)$ and $\vec{\mathsf{c}}' = (\mathsf{c}'_1, \ldots, \mathsf{c}'_n)$. For each pair of bit encryptions $(\mathsf{c}_i, \mathsf{c}'_i)$, $\widehat{\mathsf{E}}$ obtains n samples from $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}_i, \mathsf{c}'_i, \odot; U)$. Each of the n samples are either $\perp$ or an n-fold cipher. If at least a $\frac{1-\sqrt{\delta}}{2}$-fraction of the samples are not $\perp$, $\widehat{\mathsf{E}}$ continues with the first n-fold cipher as the result of the homomorphic operation; otherwise, $\widehat{\mathsf{E}}$ continues to the next pair of ciphers. If none of the pairs produced sufficiently many samples, $\widehat{\mathsf{E}}$ aborts.

We now describe emulation Steps 1-3 in the parallel emulation.

Step 1: *try at most n times to sample an n-fold encryption M_a of the string a from $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk},\mathsf{s}'}(U)$. If all n samples resulted in $\perp$, $\widehat{\mathsf{E}}$ abort.*

Step 2: *construct a circuit $C_{a,b}$ that maps a to b, and homomorphically evaluate it on M_a to obtain an n-fold encryption M_b of b. The circuit $C_{a,b}$ is constructed according to Algorithm (3), and is an amplified version of the circuit constructed in Algorithm (3).*

Algorithm 3 The amplified a to b circuit - $C_{a,b}$

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

1: sample $r_1, \ldots, r_n \leftarrow \{0,1\}^n$
2: **for** $j \in [n]$ **do**
3: obtain $\mathsf{ans}_j \leftarrow \mathbb{R}^{\triangle}_{\mathsf{s}}(r_j)$
4: obtain $\mathsf{ans}'_j \leftarrow \mathbb{R}_{a,b,\mathsf{s}}(\tilde{a} \oplus r_j)$
5: **end for**
6: **if** $\exists j \in [n]$ such that ans_j and ans'_j are not $\perp$ **then**
7: **return** $\mathsf{ans}_j \oplus \mathsf{ans}'_j$
8: **else**
9: **return** $\perp$
10: **end if**

Step 3: *transform the encryption M_b of b into b according to the procedure C_b, given by Algorithm (4), which again is an amplified version of Algorithm (2).*

Algorithm 4 The amplified b decoding procedure - C_b

Input: An encryption M of $\tilde{b}$ (allegedly $\tilde{b} = b$)
M has rows $(\vec{\mathsf{c}}_1, \ldots, \vec{\mathsf{c}}_n)$, where $\vec{\mathsf{c}}_i$ is an n-fold cipher encrypting $\tilde{b}_i$

sample $r_1, \ldots, r_n \leftarrow \{0,1\}^n$
for $j \in [n]$ **do**
  homomorphically compute from M a new matrix M' with rows $(\vec{\mathsf{c}}'_1, \ldots, \vec{\mathsf{c}}'_n)$, where $\vec{\mathsf{c}}'_i$ encrypts the i-th bit of $r'_j = \tilde{b} \oplus r_j$
  for $k \in [n]$ **do**
    set $\vec{\mathsf{c}}^*_k = (\vec{\mathsf{c}}'_1[k], \ldots, \vec{\mathsf{c}}'_n[k])$ to be the column $\mathsf{M}'[k]$, which is a 1-fold encryption of r'_j
    obtain $\mathsf{ans}_{j,k} \leftarrow \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}(\vec{\mathsf{c}}^*_k)$
    obtain $\mathsf{ans}'_{j,k} \leftarrow \mathbb{R}_{\mathsf{s}}(r'_j || \mathsf{pub}(\vec{\mathsf{c}}^*_k))$
  end for
end for
if $\exists j,k \in [n]$ such that $\mathsf{ans}_{j,k}$ and $\mathsf{ans}'_{j,k}$ are not $\perp$ **then**
  return $\mathsf{ans}_{j,k} \oplus \mathsf{ans}'_{j,k}$
else
  return $\perp$
end if

The approximated distributions. We define $d = O(1)$ distribution ensembles (or samplers) $\mathcal{D}_1, \ldots, \mathcal{D}_d$. For the extractor $\widehat{\mathsf{E}}$ to work, the circuit will be required to $(1-\delta, \mathcal{D}_i)$-approximate each of these distributions. The distributions are:

- $\mathcal{D}_{\widehat{\mathbb{A}}_{\mathsf{sk},a}}$ consists of n random strings $r_1, \ldots, r_n \in \{0,1\}^{\mathrm{poly}(n)}$ (r_i is thought of as "invoker randomness" for an n-fold encryption of the bit a_i).
- For each operation $\odot$ (from the universal set of gates) and every pair of bits (x, x'), $\mathcal{D}^{\odot,x,x'}_{\widehat{\mathbb{H}}_{\mathsf{sk}}}$ consists of n random strings $r_1, \ldots, r_n \in \{0,1\}^{\mathrm{poly}(n)}$ (for n-fold encryptions of $x \odot x'$), and random 1-fold encryptions $(\mathsf{c}, \mathsf{c}')$ 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 4.3.
- $\mathcal{D}_{\mathbb{R}_{a,b,\mathsf{s}}}$ consists of a random string $r \in \{0,1\}^n$.
- $\mathcal{D}_{\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}}$ consists of a random 1-fold encryption $(\mathsf{c}_1, \ldots, \mathsf{c}_n)$ to the bits of a random string $r \in \{0,1\}^n$.

In addition, each of the distributions $\mathcal{D}$ above also includes an index $i_{\mathbb{G}} \in [5]$, indicating which function $\mathbb{G} \in \left\{\widehat{\mathbb{A}}_{\mathsf{sk},a,\mathsf{s}'}, \widehat{\mathbb{H}}_{\mathsf{sk},\mathsf{s}'}, \mathbb{R}_{\mathsf{s}}, \mathbb{R}_{a,b,\mathsf{s}}, \mathbb{R}_{\mathsf{sk},b,\mathsf{s}}\right\}$ to invoke.

Key-independent sampling. Recall that in Definition 3.5 for d-distribution robustness, we do not require that each one of the distributions $\mathcal{D}_1, \ldots, \mathcal{D}_d$ is efficiently samplable independently of the key k. While almost all the above distributions can be sampled independently of k, the distributions of the form $\mathcal{D}^{\odot,x,x'}_{\widehat{\mathbb{H}}_{\mathsf{sk}}}$ may not be samplable without the secret encryption key sk, even given (x, x'); indeed, we only assume sampling of ciphers for random plaintexts (Definition 4.4). We do require, however, that the distributions $\{\mathcal{D}_i\}$ are jointly key-independent (Definition 3.6), implying that, for a random $i^* \leftarrow [d]$, we can sample from $\mathcal{D}_{i^*}$, independently of k (in Definition 3.6 this sampler is denoted by $\mathcal{D}^*$). In our case, this is satisfied due to random ciphertext generation: 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 using $\mathsf{RanSamp}$.

REMARK 4.1 (One uniform distribution). *In the general Definition 3.2 of unobfuscatable functions, we allow the distribution ensemble $\mathcal{D}$ to be an arbitrary samplable distribution. As mentioned in Remark 3.1, we can consider a more strict (but natural) definition where $\mathcal{D}$ is required to be the uniform distribution. Our constructions can actually achieve this notion. Indeed, the input distribution in our eventual construction is the distribution $\mathcal{D}^*$ as defined in Definition 3.6, with respect to $\mathcal{D}_1, \ldots, \mathcal{D}_d$ above. The distribution $\mathcal{D}^*$ can be made uniform.*

Specifically, the distributions $D_1, \ldots, \mathcal{D}_d$, using the symmetric key encryption described in Section 4.1, the sampler for random public parts $\mathsf{PubSamp}$ *and the sampler for random ciphers* $\mathsf{RanSamp}$ *both output uniformly random strings. So each distribution $\mathcal{D}_j$ 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 is an extra technical issue that can be taken care of: the index $i_{\mathbb{H}}$ is output 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 $i_{\mathbb{H}}$.)*

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

LEMMA 4.6. *$\mathcal{F}$ given by Construction 4.1 is non-black-box learnable in the sense of Definition 3.5.*

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

Roadmap for the proof. We analyze the success of the extractor given a circuit that successfully answers (does not output $\perp$ or errs) with probability $1 - \delta$, for each of the distributions $\mathcal{D}_1, \ldots, \mathcal{D}_d$ above, where $\delta < 2^{-5}$ is a constant. Our first step is to show the completeness of homomorphic evaluation; namely, that when the extractor performs homomorphic operations (in its parallel emulation mode), it will almost never get stuck, and will almost 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, it 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 $\nu(n)$, 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 $\nu(n) \cdot (|C| \cdot n)^{O(1)}$, which is counted within the failure probability of the extractor.

Completeness of the homomorphic evaluation phase. We call a pair of 1-fold ciphers $(\mathsf{c}, \mathsf{c}')$ encrypting a pair of bits x and x' good for $\odot$ if

$$\Pr\left[\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}(\mathsf{c}, \mathsf{c}', \odot; U) \neq \perp\right] \geq 1 - \sqrt{\delta} \ ,$$

where $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk}}(\mathsf{c}, \mathsf{c}', \odot; U)$ is as defined in Construction 4.3. We say that a pair of n-fold ciphers $(\vec{\mathsf{c}}, \vec{\mathsf{c}}')$, each encrypting a pair of bits x and x' is good for $\odot$ if there exists $i \in [n]$ such that

the 1-fold pair $(\vec{\mathsf{c}}[i], \vec{\mathsf{c}}'[i])$ 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 $\mathcal{D}^{\odot,x,x'}_{\widehat{\mathbb{H}}_{\mathsf{sk}}}$, where $(\mathsf{c}, \mathsf{c}')$ are random, with probability $1-\delta$, it holds that at least a $(1-\sqrt{\delta})$-fraction of the cipher-pairs encrypting (x, x') are good for $\odot$. Therefore, a random pair of n-fold ciphers is **not** good with probability at most $\delta^{\frac{n}{2}} \cdot d$ (indeed, the number of distributions d is a bound on the number of operations $\odot$). We will use this to show:

CLAIM 4.5. *The extractor $\widehat{\mathsf{E}}$ does not abort during the homomorphic evaluation phase, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$.*

Proof. We first claim that, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$, for every two n-fold ciphers $\vec{\mathsf{c}}$ and $\vec{\mathsf{c}}'$ that are produced 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 the 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, unlike its approximation, never outputs $\perp$), given the same inputs, output such a pair. Moreover, since $\widehat{\mathbb{H}}_{\mathsf{sk}}$ 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 $\delta^{\frac{n}{2}} \cdot d$; 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 any pair of n-fold ciphers generated by the circuit $\widehat{\mathbb{A}}^{\triangle}_{a,\mathsf{sk}}$ (or for mixed pairs created by the two circuits). The bound concrete $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$ bound is obtained by taking a union bound over at most $(|C| \cdot n)^{O(1)}$ calls made to the above oracles.

We now claim that, except with probability $2^{-\Omega(n)} \cdot (|C| \cdot n)^{O(1)}$, the extractor does not abort during the homomorphic evaluation phase. We condition on the (overwhelmingly often event) that the extractor only runs homomorphic operations for good pairs. First, note that, during Step 1, $\widehat{\mathsf{E}}$ 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 $1-\delta$, $\widehat{\mathsf{E}}$ aborts with probability at most δ^n. In any other step, 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 exists some $i \in [n]$ such that the pair of ciphers $(\vec{\mathsf{c}}[i], \vec{\mathsf{c}}'[i])$ is good for $\odot$. Therefore, each of the n 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 $1-\sqrt{\delta}$. Recall, that $\widehat{\mathsf{E}}$ aborts only if it fails to obtain more than $\frac{1-\sqrt{\delta}}{2}$-fraction of the samples in all iterations; however, this would occur in the good iteration i only with an exponentially small probability $2^{-\Omega(n)}$. Again we take a union bound over at most $(|C| \cdot n)^{O(1)}$ oracle calls. □

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 4.6. $\Pr_{C_{a,b}}[C_{a,b}(a) \neq b] \leq 2^{-\Omega(n)}$.

Proof. Recall that $C_{a,b}$, given a as input, samples n random strings $r_1, \ldots, r_n$, and queries $\mathbb{R}_{\mathsf{s}}(r_i), \mathbb{R}_{a,b,\mathsf{s}}(a \oplus r_i)$ to obtain b; this succeeds if both oracles did not output $\perp$ which is the case with probability $1-2\delta$. The probability that none of the n trials succeeds is at most $(1-2\delta)^n$. □

Completeness of the procedure C_b. We now show that the procedure C_b (almost always) returns b on the n-fold encryption of b produced by the extractor.

CLAIM 4.7. *Let $\vec{\mathsf{c}}$ be the n-fold encryption of b produced by $\widehat{\mathsf{E}}$, then*

$$\Pr_{C_b}[C_b(\vec{\mathsf{c}}) = b] \geq 1 - 2^{\Omega(n)} \cdot (|C| \cdot n)^{O(1)} \quad .$$

Proof. Throughout, we condition on the (overwhelmingly often) event that homomorphic operations do not fail, which is the cause for the $(|C| \cdot n)^{O(1)}$ factor in the final probability. Let c be a random 1-fold encryption of a random n-bit string r, then c is answered by $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ with probability at least $1-\delta$. 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. Then $(b \oplus r(\mathsf{c}), \mathsf{pub}(\mathsf{c}))$ is answered by $\mathbb{R}_{\mathsf{s}}$ with probability at least $1-\delta$; indeed, a random cipher of a random string, such as c, induces a random pair $(b \oplus r(\mathsf{c}), \mathsf{pub}(\mathsf{c}))$. It follows that with probability at least $1-2\delta$ over the choice of c both of the above queries are answered, we call such c - good.

We say that r is good if a random 1-fold encryption c of r is good with probability $1-\sqrt{2\delta}$; in particular, we know that there is a $\left(1-\sqrt{2\delta}\right)$-fraction of good r's.

Next, for a fixed r, let the matrix M be an n-fold encryption of r, 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 $(2\delta)^{\frac{n}{2}}$.

Now, 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 the i-th row of M' is an n-fold cipher encrypting the bit r'_i. Fix any such row of the matrix M', and recall that this row is generated homomorphically from a pair of ciphers $(\mathsf{c}, \mathsf{c}')$. We claim that, except with negligible probability, $\Pr\left[\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}(\mathsf{c}, \mathsf{c}', \odot; U) \neq \perp\right] \geq \frac{1-\sqrt{\delta}}{4}$; otherwise, the homomorphic evaluation procedure would skip $(\mathsf{c}, \mathsf{c}')$ with overwhelming probability. Indeed, the homomorphic procedure always evaluates the above probability, and skips any pair of ciphers whose estimate is worst than $\frac{1-\sqrt{\delta}}{2}$. 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 the i-th row of M' 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 $\frac{1-\sqrt{\delta}}{4}$-fraction of all n-fold ciphers encrypting the bit r'_i. Since this holds for every i, the matrix M' is uniformly distributed over a $\left(\frac{1-\sqrt{\delta}}{4}\right)^n$-fraction of all random cipher matrices encrypting r'.

It now follows, that for any good r', the matrix M' encrypting $r' = r \oplus b$ produced by C_b is good, except with probability $\frac{(2\delta)^{\frac{n}{2}}}{\left(\frac{1-\sqrt{\delta}}{4}\right)^n} \leq \left(\frac{\sqrt{32\delta}}{1-\sqrt{\delta}}\right)^n \leq 2^{-\Omega(n)}$, for our choice of $\delta < 2^{-5}$. Since a $\left(1-\sqrt{2\delta}\right)$-fraction of the r's are good, and our extractor tries n such r's, with overwhelming probability the extractor will go through a good matrix M' with probability $1-2^{-\Omega(n)}$. In such a case, the answers $\mathsf{ans}_{j,k}$ and $\mathsf{ans}'_{j,k}$ it obtains from the oracles $\mathbb{R}_{\mathsf{sk},b,\mathsf{s}}$ and $\mathbb{R}_{\mathsf{s}}$ are not $\perp$, and $b = \mathsf{ans}_{j,k} \oplus \mathsf{ans}'_{j,k}$ as required. □

This concludes the proof of Lemma 4.6.

□

4.4.1. 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 3.3.

LEMMA 4.7. *Robust unobfuscatable functions with unique witness unlearnable relations imply one-way functions.*

Proof. 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 $\mathcal{D} = \mathcal{U}$. (By Claim 3.4, $\frac{2}{3}$-robustness can be amplified to full robustness, so this is WLOG . Also, the proof easily generalizes to an arbitrary 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

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

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 $(\mathsf{k}, \mathsf{k}')$ 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 $\mathsf{k} \in \{0,1\}^n$:

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

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 4.2. *We note that the above lemma also holds if we consider a weaker form of robust unobfuscatable Turing machine families, whereas non-robust unobfusactable Turing machine families can be constructed without any computational assumptions (see [5]). In addition, the above also holds given a weaker inefficient extraction guarantee.*

4.5. More Efficient Extraction from Fully Homomorphic Encryption. In this section, we discuss the running time of the extractor and its effect on applications. We analyze the running of the extractor corresponding to Construction 4.1, and also present an augmented construction with more efficient extraction based on rerandomizable fully homomorphic encryption.

The extractor $\widehat{\mathsf{E}}$ given by Construction 4.3 evaluates each of the functions $\mathbb{R}^{\triangle}_{\mathsf{s}}$, $\mathbb{R}^{\triangle}_{a,b,\mathsf{s}}$, $\mathbb{R}^{\triangle}_{\mathsf{sk},b,\mathsf{s}}$, implemented by the input circuit C, $\mathrm{poly}(n)$, for some fixed polynomial poly. However, some of these evaluations are "homomorphic evaluations" of the circuit C on certain encryptions. Since each homomorphic operation is performed by evaluating the function $\mathbb{H}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$ implemented by C, we get that the total running of $\widehat{\mathsf{E}}$ is $\mathrm{poly}(n) \cdot |C|^2$. (In particular, after applying the transformation from Lemma 3.4 to obtain robust unobfuscatable functions for an arbitrary approximation parameter ε, the running time of the extractor is $\mathrm{poly}(n) \cdot (|C|/\varepsilon)^2$.)

As discussed in Section 1.4, and will be further discussed in Section 6.3.2, 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 4.1 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 perform each homomorphic operation on its own in time $\mathrm{poly}(n)$, without evaluating the entire circuit C, implementing the function $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$.

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

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

(Statistical indistinguishability can be naturally relaxed to computational.)

Known fully-homomorphic encryptions schemes from the literature (e.g., [12]) satisfy such a rerandomization property. This is usually done to guarantee function privacy, and is achieved by adding to evaluated ciphertexts a sufficient amount of noise. In our case, we would like the noise to be taken from a fixed distribution that does not depend on the evaluated circuit (in our context, the circuit we extract from); this can be ensured using the standard noise-control techniques, such as bootstrapping [25].

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. Recall that, in Construction 4.1, we required that the underlying encryption scheme is decomposable (according to Definition 4.3). We will make the same requirement for the augmented construction. We next show that, in fact, any homomorphic encryption scheme can be also made decomposable, while preserving homomorphism and rerandomization.

CONSTRUCTION 4.4 (Making 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* $c_2||r_2$ *encrypting* b_1 *and* b_2*, and an operation* $\odot$*, samples a bit* $r \leftarrow \{0,1\}$ *and outputs* $c||r$ *where the ciphertext* c *is computed homomorphically (using* Eval*) from* (c_1, c_2, r_1, r_2, r) *to encrypt* $b_1 \odot b_2 \oplus r$*.*
- *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 4.1, the algorithm* $\mathsf{PubSamp}$ *can be implemented by sampling a random cipher of a random plaintext and applying* pub*.)*

THEOREM 4.9. *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 4.5 (Robust unobfuscatable functions with efficient extraction). *The construction is a variant of Construction 4.1 with the following modifications:*

- *The encryption scheme is rerandomizable and fully homomorphic.*
- *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 4.9. The proof of black-box unlearnability is similar to the proof of Lemma 4.5. In the proof of non-black-box learnability, the main modification we introduce to the extractor given by Construction 4.3 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 $\widehat{\mathbb{H}}^{\triangle}_{\mathsf{sk},\mathsf{s}'}$). Also, the circuit C_b is given one encryption of b instead of n encryptions. C_b will now internally use the ciphertext rerandomization algorithm ReRan to generate n random encryptions of b. □

5. Publicly-Verifiable Robust Unobfuscatable Functions. In this section, we define publicly-verifiable robust unobfuscatable functions. In Section 5.1,we show 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 any candidate witness for the 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 3.2, the secret key k is required for verification.
- **Public verification of an image property.** Intuitively, vk is meant to allow verifying that a given value a is indeed the correct evaluation of f_{k} on some known input q. We generalize the latter so that vk allows to verify a given property of a candidate image a for $f_{\mathsf{k}}(q)$, for a 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 the pair (q,a) satisfies 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 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 the case that, if $a = f_{\mathsf{k}}(q)$, then $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk},q,a) = 1$.

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 $a \notin \{f_{\mathsf{k}}(q), \perp\}$.

DEFINITION 5.1 (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 $\mathsf{L} = \{\mathsf{L}_n\}_{n\in\mathbb{N}}$ and any large*

enough $n \in \mathbb{N}$*:*

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

2. **Non-Black-box learnable:** *There exists an efficient extractor* E *such that, for any function* $\varepsilon(n) \geq 2^{-\sqrt{n}}$*, 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*

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

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

$$\Pr_{\mathsf{E}}\left[(\mathsf{vk},\mathsf{z}) \in \mathcal{R}_{\mathcal{F}} : \mathsf{z} \leftarrow \mathsf{E}(C,\mathsf{vk},1^n,1^{1/\varepsilon})\right] \geq 1 - 2^{-\Omega(n)} \cdot |C|^{O(1)} \ .$$

3. **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*,* $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk},q,f_\mathsf{k}(q)) = 1$.

REMARK 5.1 (Extraction against computationally bounded circuits). *Extraction in the above definition is information theoretic, and does not require an explicit bound on the size of the circuit* C *from which extraction is made. The definition can be naturally relaxed to only support extraction from poly-size circuit families, in which case there will be some* $\mathrm{negl}(n)$ *error in the extraction probability that depends on the concrete family at hand. (See also Remark 3.4.)*

We now observe that verifiable robust unobfuscatable functions imply error-robust unobfuscatable functions.

LEMMA 5.2. *Any verifiable robust unobfuscatable function family* $\mathcal{G}$ *can be transformed into an error-robust unobfuscatable function family* $\mathcal{F}$ *(according to Definition 3.3).*

Proof. The transformation from $\mathcal{G}$ to $\mathcal{F}$: a key k' for a function $f_{\mathsf{k}'} \in \mathcal{F}$ is a pair (k,vk) consisting 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 the 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 $\mathcal{R}_{\mathcal{F}}$ is $\{((\mathsf{k},\mathsf{vk}),\mathsf{z}) : \mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk})\}$.

First, black-box unlearnability of $\mathcal{F}$ follows readily from that of $\mathcal{G}$. We show non-black-box learnability. Let C be any circuit that $(\frac{1}{2}+\varepsilon,\mathcal{D})$-approximates $f_{\mathsf{k}'}$ **even with errors** (where $\mathcal{D} = \mathcal{D}(1^n)$). The extractor $\mathsf{E}_{\mathcal{F}}$ will first obtain $O(n/\varepsilon)$ samples $(q,C(q))$, for $q \leftarrow \mathcal{D}$. It will then identify the single verification key vk such that $\mathsf{k}' = (\mathsf{vk},\mathsf{k})$; by a Chernoff bound, this can be done with probability $1 - 2^{-\Omega(n)}$, by taking the majority. $\mathsf{E}_{\mathcal{F}}$ will then construct from C a new circuit C' that $(\frac{1}{2}+\varepsilon,\mathcal{D})$-approximates f_k in the sense of Definition 5.1. This is simply done by verifying each and every answer by $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk},\cdot,\cdot)$. The completeness of $\mathsf{Ver}_{\mathcal{F}}$ says that $\mathsf{Ver}_{\mathcal{F}}$ would only filter out outputs on which C does not agree with the function g_k, implying that

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

Thus, the extractor $\mathsf{E}_{\mathcal{F}}$ can run the underlying extractor $\mathsf{E}_{\mathcal{G}}$ and obtain $\mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{vk})$ with probability $1 - 2^{-\Omega(n)} \cdot |C|^{O(1)}$. □

5.1. Constructing Verifiable Robust Families. We now show how to construct verifiable robust families from robust families with a hardcore secret (Definition 3.8), which in turn can be constructed from any robust family with a unique-witness unlearnable relation, such as the family constructed in Section 4.

THEOREM 5.3. *Assuming trapdoor permutations and non-interactive commitments, any robust unobfuscatable $\mathcal{G}$ with respect to a hardcore family $\{\mathcal{HC}_n\}$, can be compiled into a verifiable robust family $\mathcal{F}$ (according to Definition 5.1).*

Proof. We prove the theorem in two steps: we first show how to augment the key generation procedure of $\mathcal{G}$ so that it would also produce a verification key and define a corresponding (publicly-verifiable) unlearnable relation. $\mathcal{G}$ will be robust also with respect to the new relation, but still only against detectable errors. (This does not involve changing the functions in $\mathcal{G}$ themselves, but only the key generator.)

Then, in the second step, we show how to transform $\mathcal{G}$ (with its augmented generator) into a new family that also has a public-verifiable image property, and will now be robust in the stronger sense of Definition 5.1 (i.e., extraction is guaranteed whenever the image property is satisfied with high probability).

Step 1 - a family with a publicly-verifiable unlearnable relation. Given a family $\mathcal{G}$ with a hardcore family $\{\mathcal{HC}_n\}$, we define a new key generator $\mathsf{Gen}_{\mathcal{F}}(1^n)$ that samples a (private) key $\mathsf{k} \leftarrow \{0,1\}^n$ (defining a function $g_{\mathsf{k}} \in \mathcal{G}$), together with a public key $\mathsf{vk} = (h, y)$, where $h \leftarrow \mathcal{HC}_n$ is a 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 5.1. *$\mathcal{G}$ is black-box unlearnable with respect to the relation $\mathcal{R}_{\mathcal{G}}$ in the sense of Definition 5.1; namely, even given a verification key. Also, $\mathcal{G}$ is non-black-box learnable in the sense of Definition 3.2 with respect to the relation $\mathcal{R}_{\mathcal{G}}$.*[4]

Proof. First, we show that 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 verification key vk. 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 $\varphi(u)$, for an independent random u, which contradicts the one-wayness of φ.

Non-black-box learnability of $\mathcal{G}$ with respect to the relation $\mathcal{R}_{\mathcal{G}}$ follows directly from the unlearnability of $\mathcal{G}$ with respect to the hardcore family $\mathcal{HC}$ (as given by Definition 3.8. Indeed, given a circuit that $(\varepsilon, \nu, \mathcal{D})$-approximates g_{k}, and $\mathsf{vk} = (h, y)$, we can invoke the extractor given by Definition 3.8 to obtain $h(\mathsf{k})$, and thus obtain a pre-image in $\varphi^{-1}(y)$ as required.

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 5.1.

The basic idea is to embed in the function's answers rZAPs (namely, ZAPs that are resettably witness indistinguishable, see Section 6.1) attesting that the answer is consistent with a given function determined by 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 [24, 16] of using a WI proof (or rZAPs in our case) in order to get a witness-hiding proofs for statements with two independent witnesses. Details follow.

[4] Formally, non-black-box learnability in Definition 3.2 addresses the ability to learn a witness for a given privately-verifiable relation. Here we think about the natural analog: we can extract a witness with respect to the publicly-verifiable relation $\mathcal{R}_{\mathcal{G}}$.

Each function f_{k} in the family $\mathcal{F}$ consists of two independent functions g_{k_0} and g_{k_1} sampled independently from $\mathcal{G}$, randomness s for an rZAP prover, and two random strings (r_0, r_1); 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 6.1); 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 the rZAP. The length of r is chosen so to guarantee a soundness error of at most $2^{-\Omega(n)}$.

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

$$\psi(\mathsf{C}_0, \mathsf{C}_1, \mathsf{C}, q, a_0, a_1) := \left\{ \{\mathsf{C} = \mathsf{Com}(0)\} \vee \left\{ \begin{array}{c} \mathsf{C}_0 = \mathsf{Com}(\mathsf{k}_0; r_0) \\ a_0 = g_{\mathsf{k}_0}(q) \end{array} \right\} \right\} \bigwedge \left\{ \{\mathsf{C} = \mathsf{Com}(1)\} \vee \left\{ \begin{array}{c} \mathsf{C}_1 = \mathsf{Com}(\mathsf{k}_1; r_1) \\ a_1 = g_{\mathsf{k}_1}(q) \end{array} \right\} \right\} \ ;$$

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 5.2. *The function family $\mathcal{F}$ is verifiable and robust according to Definition 5.1.*

Proof. 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 $\varepsilon = \varepsilon(n)$ can be transformed into a learner L' for the family $\mathcal{G}$.

Given oracle access to a random g_{k}, and its verification key vk', L' samples a random bit b on it own, and treats its oracle g_{k} as g_{k_b} and the verification key vk' as vk_b. In addition, L' samples $g_{\mathsf{k}_{1-b}} \in \mathcal{G}$, 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 the answer a_{1-b}, and the oracle for the answer a_b. The proof is then given using what we referred above as "the first witness"; namely, for the first part of the "and statement" b and the randomness for $\mathsf{C} = \mathsf{Com}(b)$ are 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 $\varepsilon/2 - \mathrm{negl}(n)$, 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, we can first move to an hybrid experiment where C_b is a commitment to k_b, and then using the fact that the rZAP is (resettably) witness-indistinguishable, we can move to another hybrid experiment where "the second witness" for the WI statement is used. In this experiment 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 would still succeed with probability $\varepsilon - \mathrm{negl}(n)$. In this experiment, the view of L is independent of b, and so it must output the right z_b at least in $\frac{1}{2}$ of its successes. The claim follows.

We now show that $\mathcal{F}$ is non-black-box learnable according to Definition 5.1. 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 $\psi(\mathsf{C}_0, \mathsf{C}_1, \mathsf{C}, q, a_0, a_1)$, where $(\mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$ are determined by vk.

By the completeness of the rZAP, $\mathsf{Ver}_{\mathcal{F}}$ has the required completeness: $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, (q, r), f_{\mathsf{k}}(q, r)) = 1$ for any triple. Next, for $\varepsilon \geq 2^{-\sqrt{n}}$, let C be any circuit that $(\varepsilon, \widehat{\mathcal{D}})$-approximates f_{k} in the weak sense given by Definition 5.1. 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, with probability $1 - 2^{-\Omega(n)}$, one of them will $(\varepsilon, \nu, \mathcal{D})$-approximate the underlying function g_{k_0} or g_{k_1}, for $\nu = 0$; namely, with no errors. The circuit C_b will have a random first rZAP message r hardwired into it. Given a sample q from $\mathcal{D}(1^n)$, C_b 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 determined by the commitment C given in the verification key vk. By the soundness of the rZAP, with probability $1 - 2^{\Omega(n)}$ over the choice of randomness r, C_b $(\varepsilon, 0, \mathcal{D})$-approximates g_{k_b}. Hence, it suffices that the extractor $\mathsf{E}_{\mathcal{F}}$ feeds each one of (C_0, C_1) to the extractor $\mathsf{E}_{\mathcal{G}}$ of the family $\mathcal{G}$, and it will manage to produce $\mathsf{z} \in \mathcal{R}_{\mathcal{G}}(\mathsf{k}_b)$ with the required probability. □

REMARK 5.2 (Assuming non-interactive commitments). *In the above, we rely on non-interactive commitments. These are known based on any keyed family of one-to-one one-way functions (in particular, trapdoor permutations) that have a recognizable key space; namely it is possible to verify that the function represented by a give key is indeed injective. Such recognizability is crucial to guarantee binding. (A fixed, keyless, one-to-one would naturally also do.) In our application above, we can assume a weak recognizability property: injectiveness can be only* privately *testable, given the random coins used to generate the key. Indeed, the circuit evaluating the unobfuscatable functions, in particular, provides a (statistical) ZAP proof for the validity of the commitment, and thus can prove that it chose the key properly.*

This concludes the proof of Theorem 5.3 □

6. From Verifiable Robust Unobfuscatable Functions to Resettable Protocols. In this section, we apply verifiable robust unobfuscatable functions in order to construct new resettable protocols.

6.1. Definitions. We briefly recall the definitions of resettable soundness [38, 4], resettable ZK [13], and several other basic definitions and tools used in this section. Most definitions are taken almost verbatim from [13, 4, 20].

Resettably-soundness. We briefly recall the definitions of resettable soundness presented in [4]. 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 [4], 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 6.1 (Resettably-sound argument [4]). *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 $t = \mathrm{poly}(n)$ 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 (P, V) is a resettably-sound argument for $\mathcal{L}$ if, 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 arguments of knowledge. In a resettably-sound argument of knowledge, we can efficiently extract a witness from any resetting prover that convinces the verifier of accepting with noticeable probability. We shall further restrict attention to such arguments where the knowledge extractor uses the prover as a black-box, except for being given its size.

DEFINITION 6.2 (Resettable argument of knowledge). *A protocol (P, V) is a resettable argument of knowledge if there exists a PPT (black-box) extractor Ext such that, for any poly-size resetting prover P^*, and all large enough $n \in \mathbb{N}$ and $x \in \{0,1\}^n$: if $(\mathsf{P}^*, \mathsf{V})(x) = 1$ with probability at least $\epsilon(n)$, then $\mathsf{Ext}^{\mathsf{P}^*}(x, |\mathsf{P}^*|)$ outputs some $w \in \mathcal{R}_{\mathcal{L}}(x)$ with probability $\mathrm{poly}(\epsilon/|\mathsf{P}^*|)$.* [5]

Resettably-sound ZK and witness-indistinguishable arguments. A resettably-sound ZK (or WI) argument satisfies the above definition of resettable-soundness (or argument-of-knowledge), and is ZK (or WI) in the usual sense. To obtain resettably-sound ZK arguments of knowledge (rsZKAOK) we shall rely on resettably-sound WI arguments of knowledge (rsWIAOK) as a tool. rsWIAOK systems can be constructed from classic public-coin proof systems such as (the n-fold version of) Blum's Hamiltonicity WI protocol [11] by applying to them the [4] transformation (the resulting constructions are based solely on one-functions). These classic protocols indeed have a knowledge extractor that uses the prover only as a black-box (except for being given its size).

Resettable ZK. we start by recalling resettable ZK.

DEFINITION 6.3 (Resettable zero-knowledge [13]). *An interactive proof system (P, V), for a language $\mathcal{L}$, is said to be resettable zero-knowledge if, for every poly-size 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^*

[5] In the literature [8] a tighter definition is sometimes considered where Ext outputs a witness with probability 1 in expected time $\frac{\mathrm{poly}(n)}{\epsilon(n) - \mathrm{negl}(n)}$.

- *Randomly select and fix $t = \text{poly}(n)$ 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 $\mathcal{S}(\mathsf{V}^*, \vec{x})$.

Again for simplicity, we concentrate on the simple case of one prover incarnation; however, all of our results directly extend to the model of multiple incarnations.

Resettable ZAPs. ZAPs are two-message public-coin witness-indistinguishable proofs introduced by Dwork and Naor [21]. They further have the special property that the first message (sent by the prover) can be reused for multiple proofs. As noted in [4], any ZAP already has the property of resettable soundness. Furthermore, resettable witness-indistinguishability property can be obtained by applying the transformation in [13]. We refer to the resulting system as an rZAP system having the property of resettable soundness as well as resettable-witness indistinguishability.

Commitments. We use two types of perfectly (or statistically) binding commitments. One type is non-interactive commitments, which can be constructed based on injective one-way functions with a (privately) recognizable key space (see Remark 5.2). The second are interactive (two-message) commitments that can be constructed from one-way functions [39].

6.2. 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 $\mathcal{R}_{\mathcal{F}}$ and input distribution $\mathcal{D}$. Let $\mathsf{Gen}_{\mathcal{F}}, \mathsf{Ver}_{\mathcal{F}}$ be the key generation algorithm and the image verification relation of $\mathcal{F}$. The protocol is detailed in Figure 6.1.

Protocol 6.1

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 a rsWIAOK:
   "$x \in \mathcal{L}$ or $\{\exists \mathsf{z} : (\mathsf{vk}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}}\}$".

Fig. 6.1: A resettably-sound (concurrent) ZK protocol

6.3. Security Analysis. In this section, we analyze the resettable security of Protocol 6.1, according to the different settings of the parameter m.

6.3.1. Resettable Soundness. We show that Protocol 6.1 is resettably-sound, for any setting of m (constant or $\mathrm{poly}(n)$).

LEMMA 6.4. *For any $1 \leq m \leq \mathrm{poly}(n)$, Protocol 6.1 is a resettably-sound ZK argument of knowledge.*

Proof. Let P^* be a poly-size (resetting) prover that, for a set $\mathcal{X}$ of inputs, convinces V to accept each $x \in \mathcal{X} \cap \{0,1\}^n$ with probability $\varepsilon(n)$. We describe a PPT extractor E that, given black-box access to P^* extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$ with probability $\mathrm{poly}(\varepsilon(n))$, for all but finitely many $x \in \mathcal{X}$. First, we consider a new prover P_1^* against the rsWIAOK protocol executed in the last step of the protocol. 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^*, by 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 knowledge extraction guarantee, we know that E_{wi} outputs a witness for the WI statement corresponding to x and vk, with probability $\mathrm{poly}(\varepsilon)$. 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 E_{wi} and P_1^* to break the unlearnability of $\mathcal{F}$. Indeed, since E_{wi} is a black-box extractor (see Section 6.1), it can answer all of P_1^*'s calls to the function, using an external oracle to f_{k}. □

6.3.2. Stand-Alone (Non-Resettable) ZK. In this section, we show that Protocol 6.1 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 the one used in Protocol 6.1, 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 bounded by $\mathrm{poly}(n, |C|) \cdot \varepsilon^{-d}$. We will show that the protocol is resettably-sound when $m \geq d$.

At high-level, the correlation between d and m is required to ensure that our simulator would be able to run in expected poly-time. Here, the problematic case is when the probability $\varepsilon(n)$ that the verifier finishes the execution is neither noticeable or negligible. In such a case, to make sure that the simulated distribution is not skewed, the simulator should complete the execution with roughly the same probability $\varepsilon = \varepsilon(n)$ as the verifier. The problem is that in such a case naively trying to extract from a single slot would take time $\mathrm{poly}(n, \mathsf{V}^*) \cdot \varepsilon^{-d}$, so overall the simulator's running time would be roughly $\mathrm{poly}(n, \mathsf{V}^*) \cdot \varepsilon^{-d+1}$, which may not be polynomially bounded.

The addition of extra slots allows to ensure that with high-probability in one of these slots the probability that the verifier finishes the slot is roughly $\varepsilon^{\frac{1}{d}}$. Our simulator can thus first estimate the probability ε, and then make repeated trials to find a circuit that $\varepsilon^{\frac{1}{d}}$ approximates the unobfuscatable function and extract the trapdoor from this slot.

LEMMA 6.5. *Protocol 6.1 with $m \geq d$ is ZK.*

Proof. Similarly to the simulator of Goldreich and Kahan [27], our simulator starts by running the cheating verifier V^* once, honestly simulating its view until the proof step. We

say that V^* aborts if in some slot it does not answer or returns an answer a to a query q that does not verify under the verification key vk it sent. If V^* does not abort and evaluates the function correctly in each one of the m slots, the simulator interacts with V^* repeatedly to estimate the probability ε that V^* evaluates the function correctly in all the slots. Specifically it samples executions of the protocol until it obtains n successes and then sets the estimate as $\tilde{\varepsilon} = n/N$, where N is the number of trials.

Using a standard tail bound, it can be seen that, except with probability $2^{-\Omega(n)}$, it holds that $\tilde{\varepsilon} \in (\frac{\varepsilon}{2}, 2\varepsilon)$; throughout the rest of the proof, we assume that this is indeed the case.

The next step of the simulator is to obtain a slot that computes the function f_{k} with high-enough probability. For simplicity of notation we assume that the protocol has exactly $m = d$ slots, however the same argument holds also for $m > d$. Let $S_1, \ldots, S_d$ be random variables describing each of the d slots, and let G_i be the event that the verifier does not abort in the i-th slot, and let $\neg H_i$ be the event that $S_1, \ldots, S_{i-1}$ are such that the verifier does not abort in S_i with probability at least $(\varepsilon/2)^{1/d}$, i.e. $\Pr[G_i \mid S_1, \ldots, S_{i-1}] \geq (\varepsilon/2)^{1/d}$. The simulator is going to try and sample $S_1, \ldots, S_{i-1}$ such that $\neg H_i$ holds and then use the residual circuit for the slot S_i to extract the key. To do so, our simulator will repeatedly sample protocol executions until it obtains n independent protocol executions $S_1, \ldots, S_d$ that are non aborting; namely, such that $G_1, \ldots, G_d$ hold. For each of these n executions, the simulator would create d residual circuits, for each one of the slots, and run the extractor of $\mathcal{F}$, with approximation parameter $(\tilde{\varepsilon}/4)^{1/d}$, provided that $(\tilde{\varepsilon}/4)^{1/d} \geq 2^{-\sqrt{n}}$; if this is not the case, the simulator aborts, which has a negligible $2^{-\Omega(\sqrt{n})}$ statistical effect on the simulated distribution.

We claim that, except with probability $2^{-\Omega(n)}$, the simulator does not "get stuck"; that is, it obtains the trapdoor from one of the $n \cdot d$ circuits. Correctness of the simulation then follows from the WI property of rsWIAOK. We now prove this, and then analyze the running time of the simulator.

First, we claim that conditioned on sampling a non-aborting execution $S_1, \ldots, S_d$, the event $\neg H_i$ occurs for some i, with probability at least $\frac{1}{2}$ over the choice of $S_i | S_1, \ldots, S_{i-1}$. Indeed, by the residual probability Claim 2.1, it holds that $\Pr[G_1, \ldots, G_d, H_1, \ldots, H_d] \leq \left((\varepsilon/2)^{1/d}\right)^d = \varepsilon/2$, when setting $k = m = d$, and $\delta = (\varepsilon/2)^{1/d}$, and thus

$$\begin{aligned} \Pr[\exists i : \neg H_i \mid G_1, \ldots, G_d] = \\ 1 - \Pr[H_1, \ldots, H_d \mid G_1, \ldots, G_d] = \\ 1 - \frac{\Pr[G_1, \ldots, G_d, H_1, \ldots, H_d]}{\Pr[G_1, \ldots, G_d]} \geq \\ 1 - \frac{\varepsilon/2}{\varepsilon} \ . \end{aligned}$$

It follows that, except with probability 2^{-n}, the simulator indeed finds a circuit satisfying $\neg H_i$. When it runs the extractor with approximation parameter $(\tilde{\varepsilon}/4)^{1/d} \leq (\varepsilon/2)^{1/d}$, it will manage to extract the key k with probability $1 - 2^{-\Omega(n)} \cdot |\mathsf{V}^*|^{O(1)}$.

It is left to analyze the simulator's running time. The simulator simulates an execution only if the initial transcripts was accepting, which happens with probability ε. Hence, it suffices to show that the expected running time of the rest of the simulation is bounded by $\mathrm{poly}(n)/\varepsilon$. Indeed, estimating $\tilde{\varepsilon}$ takes expected time n/ε. Then rejection sampling for each non-aborting execution $S_1, \ldots, S_d$ takes time $1/\varepsilon$, and overall n/ε. Then, for each of the $n \cdot d$ circuits, the extractor runs in expected time $\mathrm{poly}(n, |\mathsf{V}^*|) \cdot \left(\varepsilon^{1/d}\right)^{-d}$. Thus, the overall running time is bounded as required by $\varepsilon \cdot \mathrm{poly}(n)/\varepsilon = \mathrm{poly}(n)$. □

Round-efficient resettably-sound ZK. As explained in Section 4.5, the construction described in Section 4 either yields extractors with running time $\mathrm{poly}(n) \cdot (|C|/\varepsilon)^2$, assuming trapdoor permutations, or extractors running in time $\mathrm{poly}(n) \cdot |C|/\varepsilon$, assuming fully-homomorphic encryption. Plugging this into Protocol 6.1, we would get by Lemma 6.5, an eight-message protocol in the first case and a six-message one in the second case, if we use say a three-message rsWIAOK (e.g. based on [11]). 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 2) similarly to the protocol of [23].

COROLLARY 6.6 (of Lemma 6.5). *Assuming trapdoor permutations there exists a six-message resettably-sound ZK protocol.*

COROLLARY 6.7 (of Lemma 6.5). *Assuming fully homomorphic encryption and trapdoor permutations, there exists a four-message resettably-sound ZK protocol.*

6.3.3. Concurrent and Resettable ZK. In this section, we show that for $m = n^{\Omega(1)}$ Protocol 6.1 is concurrent ZK. A simultaneously-resettable ZK protocol can then be obtained from Protocol 6.1 and any simultaneously-resettable WI protocol by applying a general transformation of Deng, Goyal, and Sahai [20].

LEMMA 6.8. *For every constant $\delta > 0$, Protocol 6.1 with $m = n^{\delta}$ is concurrent zero-knowledge.*

Proof. We start with an overview of the simulator.

Overview of the simulation. The simulation uses the techniques of Richardson and Kilian [48], and more specifically, the slightly augmented version of them applied in [20, 15]. We start by describing the main ideas behind this technique as they were 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 meant 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 the look-ahead threads and therefore in 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 actually 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, construct smaller simulation circuits themselves and extract from them.

We now describe the basic 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 2 where V^* evaluates the function f_{k} as a slot. In a single session, there are n^{δ} slots, and thus 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 randomized 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 $\mathcal{C}$ 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. $\mathcal{C}$ 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 $\mathcal{C}$ in the same way that they are used in the MAIN function.

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

We note the 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 value of m.
- The function EXTRACT uses the circuit $\mathcal{C}$ generated by the function EMULATE.
- The circuit $\mathcal{C}$ generated by the function EMULATE simulates messages according to the strategy of the function MAIN. In particular, $\mathcal{C}$ 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 V^* 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 verification key 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 $(m+1)$-st slot, MAIN terminates and outputs the current value of T.

The function EMULATE. On parameters (T, m), the function creates a circuit $\mathcal{C}$ and outputs it. $\mathcal{C}$ 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 $\mathcal{C}$ and the function MAIN is that $\mathcal{C}$ gets 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 ends, $\mathcal{C}$ 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, $\mathcal{C}$ outputs $\perp$. As described above, the circuit $\mathcal{C}$ is a randomized circuit; before outputting $\mathcal{C}$, EMULATE samples random coins and hard-codes them into $\mathcal{C}$; thus, eventually the output of EMULATE is a deterministic circuit. We note that the circuit $\mathcal{C}$ emulates uniform computations, and thus when creating it, we need to apriori bound the number of steps it will emulate. This bound will be a polynomial determined according to (T, m), and is analysed below as part of the run-time analysis.

The function EXTRACT. On parameters (T, m), the function calls the function EMULATE with the same parameters and obtains a circuit $\mathcal{C}$. EXTRACT then executes the extractor E of the unobfuscatable functions family on $\mathcal{C}$ with threshold $\varepsilon = \frac{1}{16}$ (see Definition 5.1), and outputs the same as E.

Correctness of the simulation. Let $\mathsf{Real}(x, w)$ be the view of V^* in a real interaction $(\mathsf{P}(w), \mathsf{V}^*)(x)$, 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

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

It is thus enough to show that

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

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 mental experiment: during simulation, before executing the extractor on some circuit, we execute this circuit several times (using independent randomness) to see if the execution of this circuit is likely to get stuck. The simulation performed by the circuit itself will recursively follow the same augmented behavior. If in any of the executions the circuit get stuck, we do not even try to extract, but 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 $\mathcal{C}$, and executes the extractor E of the unobfuscatable functions family on $\mathcal{C}$. FORK then repeats the following n independent times: sample $q \leftarrow \mathcal{D}(1^n)$ and run $\mathcal{C}(q)$. We shall refer to the computation transcript produced by $\mathcal{C}$ 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 $\mathcal{C}$ constructed by EMULATE makes a call to FORK that gets stuck, $\mathcal{C}$ outputs $\perp$.

Finally, FORK outputs the same as E on $\mathcal{C}$ (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 exists 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 $p_1(n)$.

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 $p_2(n)$. 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 the event that $\mathsf{Bad}_2 \wedge \neg H_i$. 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 $p_3(n)$.

LEMMA 6.9. $\Pr[\{(|\{i \in [2\Delta]|G_i\}| \geq \Delta\} \wedge \{H_1 \wedge \ldots \wedge H_{2\Delta}\}] \leq \text{negl}(n)$

Proof. By the residual probability Claim 2.1, when setting $m = 2\Delta, k = \Delta, \delta = \frac{1}{8}$, and fixing any set $\{i_j\}_{j\in[\Delta]} \subset [2\Delta]$, we have:

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

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

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

□

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 occur 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 does not occur 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 $\mathcal{C}(q)$, performed by FORK starting from state T, and outputting 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 $\mathcal{C}$ with $\varepsilon = \frac{1}{16}$, 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:

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

and therefore

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

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 function 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 6.8 showing that Protocol 6.1 is concurrent zero-knowledge.

□

6.4. Resettable Protocols 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 first corollary, by plugging our protocol into the transformation of [4], we get an $\tilde{O}(\log n)$-round resettable ZK argument of knowledge based on one-way functions. As a second corollary, when combining our protocol with the simultaneously-resettable WI from one-way functions [18], and a general transformation of [20] (as augmented by [18]), we obtain a *simultaneously-resettable ZK protocol from one-way functions*.

THEOREM 6.10 (resettably-sound (concurrent) ZK from one-way functions). *Protocol 6.2 with any $m \geq 2$ is a resettably-sound ZK protocol, and with $m = n^{\Omega(1)}$, it is a resettably-sound concurrent ZK protocol.*

COROLLARY 6.11 (of Theorem 6.10 and [4]). *There exists a $\tilde{O}(\log n)$-round resettable ZK argument of knowledge based on one-way functions.*

COROLLARY 6.12 (of Theorem 6.10, [20], and [18]). *There exists a simultaneously-resettable ZK argument of knowledge based on one-way functions.*

Corollary 6.11 follows by plugging in our one-way function based resettably-sound ZK protocol into the transformation of [4], and Corollary 6.11 follows by plugging in a simultaneously-resettable WI protocol from one way functions [18] into the transformation of [20]. (More accurately, the transformation of [20] is described for ZAPs, but as observed in [18] it actually holds for any simultaneously-resettable WI.)

The augmented protocol. The protocol follows similar ideas to those used for making robust unobfuscatable functions (with a hardcore secret) verifiable, as presented in Section 5.1. Specifically, non-interactive commitments are replaced with two-message commitments. ZAPs are replaced with instance-dependent rWIs (see Definition 6.13 below), where the dependance is on the proven statement x.

Before describing and analysing the protocol, we define instance-dependent resettable witness-indistinguishability.

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 $\mathcal{L}'$. The soundness and rWI properties are decoupled according to y: if $y \in \mathcal{L}'$ the system is sound and if $y \notin \mathcal{L}'$ the system is rWI. More specifically, we require that from any resetting verifier that breaks the WI property, we can extract a witness for $y \in \mathcal{L}'$. (As in the AOK Definition 6.2, we consider a definition in which the extractor runs in strict poly-time and the extraction probability may drop polynomially.)

DEFINITION 6.13 (rWI^y). *An argument system* (P, V) *for* $\mathcal{L}$ *is rWI with respect to* $\mathcal{L}'$ *if it satisfies:*

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

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

2. **Instance-dependent rWI:** *there exists a PPT extractor* E *such that for any poly-size resetting verifier* V^**, all large enough* $x \in \mathcal{L} \cap \{0,1\}^n$*,* $w_x^{(1)}, w_x^{(2)} \in \mathcal{R}_{\mathcal{L}}(x)$*, and* $y \in \{0,1\}^n$*, and for every function* ϵ*:*

$$\textit{if } \Pr\left[\left(\mathsf{P}(w_x^{(1)}), \mathsf{V}^*\right)(x,y) = 1\right] - \Pr\left[\left(\mathsf{P}(w_x^{(2)}), \mathsf{V}^*\right)(x,y) = 1\right] \geq \epsilon(n) \ ,$$
$$\textit{then } \Pr_{\mathsf{E}}\left[w_y \leftarrow \mathsf{E}^{\mathsf{V}^*}(x, y; w_x^{(1)}, w_x^{(2)}) \;\middle|\; w_y \in \mathcal{R}_{\mathcal{L}'}(y)\right] \geq \epsilon(n) - \mathrm{negl}(n) \ .$$

Constructing instance-dependent rWIs from one-way functions. Instance-dependent rWI can be constructed based on the Canetti-Goldreich-Goldwasser-Micali transformation [13], and instance-dependent commitment schemes (also known as trapdoor commitments or chameleon commitments), which in turn can be constructed from one-way functions [23]. We briefly recall what are the properties of such commitments (extending [23, Definition 4.1]).

An instance-dependent commitment scheme is 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 decommitment information are computationally indistinguishable from honestly generated commitments and their decommitment information.
2. **Instance-dependent binding:** Given the instance y, for any commitment transcript (including all messages of the commitment phase), and any two valid decommitments to two distinct values, it is possible to efficiently compute a witness $w \in \mathcal{R}_{L'}(y)$.

Given such commitments, it is possible to transform any public-coin WI system (e.g. Blum [11]) to an instance-dependent rWI, using the [13] 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 $y \notin \mathcal{L}$, the commitment is binding, and rWI follows as in [13].

We proceed with the description of the augmented protocol. Let $\mathcal{G}$ be a family of robust unobfuscatable functions with respect to a hardcore family $\{\mathcal{HC}_n\}$ (as given by Definition 3.8, and constructed in Sections 3 and 4 based on one-way functions). Let φ be a one-way function. We make use of a two-message statistically binding commitment Com (e.g., [39]). 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 a resettably-sound witness-indistinguishable arguments of knowledge rsWIAOK (see Definition 6.13 and definitions in Section 6.1). Similarly to Protocol 6.1, m is a parameter that controls the number of slots. We set $m = 2$ to get constant round resettably-sound ZK and $m = n^{\Omega(1)}$ to get a resettably-sound concurrent ZK.

Protocol 6.2

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 V.
2. V samples a bit $b \leftarrow \{0,1\}$.
   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 a randomly chosen hardcore function, and $y_i = \varphi(h_i(\mathsf{k}_i))$.
   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:

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

   V uses the randomness of C_0 and C_1 as a witness.
4. P proves the following statement to V using an rsWIAOK:
   "$x \in \mathcal{L}$ or $\{\exists \mathsf{z} : y_0 = \varphi(\mathsf{z}) \vee y_1 = \varphi(\mathsf{z})\}$".

Fig. 6.2: A resettably-sound (concurrent) ZK protocol from any one-way function

Theorem 6.10 follows from the following lemmas:

LEMMA 6.14. *For any $1 \leq m \leq \mathrm{poly}(n)$, Protocol 6.2 is a resettably-sound argument of knowledge.*

LEMMA 6.15. *Let d be a constant such that the running time of the non-black-box extractor of $\mathcal{G}$ (the robust unobfuscatable functions family used in Protocol 6.2), given security parameter n, and approximation parameter ε, is bounded by $\mathrm{poly}(n, |C|) \cdot \varepsilon^{-d}$. Then Protocol 6.2 with $m \geq d$ is (stand-alone) ZK.*

LEMMA 6.16. *For every constant $\delta > 0$, Protocol 6.2 with $m = n^{\delta}$ is concurrent zero-knowledge.*

Proof of Lemma 6.14. The proof of the lemma closely follows the proofs of Lemma 6.4

and Theorem 5.3. Let P^* be a poly-size prover that, for a set $\mathcal{X}$ of inputs, convinces V to accept each $x \in \mathcal{X} \cap \{0,1\}^n$ with probability $\varepsilon(n)$. We describe a PPT extractor E that, given black-box access to P^* extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$ with probability $\mathrm{poly}(\varepsilon(n)) - \mathrm{negl}(n)$.

We start by considering a sequence of hybrid experiments. For every $i \in [m]$, in the i-th experiment P^* interacts with a verifier $\mathsf{V}_{1,i}$ that is defined just as V except that in the first i executions of the rWI^x protocol, $\mathsf{V}_{1,i}$ proves the statement uses the decommitment information for C and C_{1-b} as a witness instead of C_0 and C_1.

If the probability that P^* convinces $\mathsf{V}_{1,m}$ to accept x with probability less than $\frac{\varepsilon(n)}{2}$, then by the instance-dependent rWI property, there exist an extractor E_1 that extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$ from P^* (acting as the rWI verifier) with probability at least $(\frac{\varepsilon(n)}{m} - \frac{\varepsilon(n)}{2m}) - \mathrm{negl}(n)$.

For the rest of the proof we thus assume that P^* convinces $\mathsf{V}_{1,m}$ to accept x with probability at least $\frac{\varepsilon(n)}{2}$. We next consider another hybrid experiment where P^* interacts with a verifier V_2 that is defined just as $\mathsf{V}_{1,m}$ except that V_2 samples a random bit $b \in \{0,1\}$ and sets C_b to be a commitment to 0^n instead of a commitment to k_b. By the computational hiding property of the commitment, P^* convinces $\mathsf{V}_{1,m}$ to accept x with probability at least $\varepsilon_1(n) = \frac{\varepsilon(n)}{2} - \mathrm{negl}(n)$.

Next, we consider a new prover P_1^* against the rsWIAOK protocol. The new prover P_1^* is given oracle access to a random $g_{\mathsf{k}} \in \mathcal{G}$, as well as a pair (h, y) for a random hardcore function $h \leftarrow \mathcal{HC}_n$ and $y = \varphi(h(\mathsf{k}))$. P_1^* samples $b \leftarrow \{0,1\}$, treats its oracle g_{k} as g_{k_b} and sets $\mathsf{vk}_b = (h_b, y_b) = (h, y)$. P_1^* samples $g_{\mathsf{k}_{1-b}} \in \mathcal{G}$ and $\mathsf{vk}_{1-b} = (h_{1-b}, y_{1-b})$, where $h_{1-b} \leftarrow \mathcal{HC}_n$ and $y_i = \varphi(h_i(\mathsf{k}_i))$ as in Protocol 6.2. P_1^* tries to convince a rsWIAOK verifier of the WI statement corresponding to x and y_0, y_1 (defined in the last step of Protocol 6.2). To do so P_1^* emulates P^*. First, P_1^* receives from P^* the first message r of a statistically binding commitment. Then, P_1^* obtains the commitments $\mathsf{C} = \mathsf{Com}_r(b)$, $\mathsf{C}_b = \mathsf{Com}_r(0^n)$ and $\mathsf{C}_{1-b} = \mathsf{Com}_r(\mathsf{k}_{1-b})$ and sends $(\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$ to P^*. When P^* sends a query q, P_1^* computes on its own $a_{1-b} = g_{\mathsf{k}_{1-b}}(q)$ and obtains $a_b = g_{\mathsf{k}_b}(q)$ by making an oracle query to g_{k_b}. P^*'s messages in the rWI^x proof are answered using the decommitment information for C and C_{1-b} as a witness. P^*'s messages in the rsWIAOK proof are forwarded to the external verifier. P_1^* perfectly emulates interaction of P^* with V_2, and thus convinces the external verifier with probability at least $\varepsilon_1(n)$.

We can now apply the black-box extractor E_2 for the rsWIAOK to extract a witness from P_1^*. To do so, E_2 can sample g_{k} and (h, y) on its own, and use them to answer any oracle call that P_1^* makes. By the AOK guarantee, we know that E_2 outputs a witness for the WI statement corresponding to x and y_0, y_1 (in expected polynomial time) with probability at least $\varepsilon_2(n) = \mathrm{poly}(\varepsilon_1(n)) - \mathrm{negl}(n)$. If with probability at least $\varepsilon_2/2$, we extract $w \in \mathcal{R}_{\mathcal{L}}(x)$ rather than z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$, then we are done.

Thus, from hereon we assume the probability E_2 outputs z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$ is at least $\frac{\varepsilon_2(n)}{2}$. First, we claim that E_2 outputs z such that $y_b = \varphi(\mathsf{z})$ only with negligible probability. Otherwise, by the unlearnability property of $\mathcal{G}$, E_2 could find such z given only black-box access to g_{k} and $y = \varphi(u)$ for a random u contradicting the one-wayness of φ. Therefore, E_2 must output z such that $y_{1-b} = \varphi(\mathsf{z})$ with probability at least $\varepsilon_3(n) = \frac{\varepsilon_2(n)}{2} - \mathrm{negl}(n)$.

Next, we consider a prover P_2^* that is defined exactly as P_1^* except that the P_2^* samples both $g_{\mathsf{k}_0}, g_{\mathsf{k}_1}$ and $\mathsf{vk}_0, \mathsf{vk}_1$ itself, and emulates P^* using a commitment $\mathsf{C}_b = \mathsf{Com}_r(\mathsf{k}_b)$ instead of $\mathsf{C}_b = \mathsf{Com}_r(0^n)$. Now, since E_2 is a black-box extractor, it follows from the computational hiding property of the commitment that E_2 given access to P_2^* still outputs z such that $y_{1-b} = \varphi(\mathsf{z})$ with probability at least $\varepsilon_3(n) - \mathrm{negl}(n)$.

Next, for every $i \in [m]$, we consider a prover $\mathsf{P}_{3,i}^*$ that is defined exactly as P_2^* except that it answers P^*'s messages in the first i executions of the rWI^x proof using the decommitment

information for C and C_b as the witness, instead of using the opening of C and C_{1-b}. To conclude the proof we show that E_2 given access to $\mathsf{P}^*_{3,m}$ outputs z such that $y_{1-b} = \varphi(\mathsf{z})$, with probability $\varepsilon_4(n) \leq \mathrm{negl}(n)$. Then, since E_2 is a black-box extractor and by the instance-dependent rWI property, there exist an extractor E_3 that extracts a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$ with probability at least $\frac{\varepsilon_3(n)}{2m} - \mathrm{negl}(n)$.

To prove that $\varepsilon_4(n) \leq \mathrm{negl}(n)$, we consider a prover P^*_4 that is defined exactly as $\mathsf{P}^*_{3,m}$ except that P^*_4 samples g_{k_b} and vk_b itself, while $g_{\mathsf{k}_{1-b}}$ is accessed as an oracle, and $\mathsf{vk}_{1-b} = (h_{1-b}, y_{1-b})$ is given as input. Additionally, P^*_4 emulates P^* using a commitment $\mathsf{C}_{1-b} = \mathsf{Com}_r(0^n)$ instead of $\mathsf{C}_{1-b} = \mathsf{Com}_r(\mathsf{k}_{1-b})$. Now, since E_2 is a black-box extractor, it follows from the computational hiding property of the commitment that E_2 given access to P^*_4 outputs z such that $y_{1-b} = \varphi(\mathsf{z})$ with probability at least $\varepsilon_4(n) - \mathrm{negl}(n)$. However, by the unlearnability property of $\mathcal{G}$, it follows that E_2 could find such z given only black-box access to $g_{\mathsf{k}_{1-b}}$ and $y_{1-b} = \varphi(u)$ for a random u, thus inverting the one-way φ with probability $\varepsilon_4 - \mathrm{negl}(n)$. Therefore, $\varepsilon_4(n) \leq \mathrm{negl}(n)$ as required. □

Proof of Lemma 6.15. The simulator $\mathcal{S}$ starts by running the cheating verifier V^* once, honestly simulating its view until the proof step. Let r be the first message sent in this first execution and let $(\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$, where $\mathsf{vk}_i = (h_i, y_i)$ for $i \in \{0,1\}$, be V^*'s response. Note that with overwhelming probability over r, the commitment Com_r is perfectly binding. In the rest of the proof we assume that this is indeed the case and fix the first simulator message to r in all subsequent simulated executions.

We say that V^* aborts if in some slot it does not answer one of the messages, or if the rWI^x proof of the slot is not accepting. If V^* does abort, $\mathcal{S}$ simply output the view of V^* in this initial execution. If V^* does not abort, then $\mathcal{S}$ continues to interacts with V^* repeatedly to estimate the probability ε that V^* does not abort (conditioned on the fixed first message r). Specifically, $\mathcal{S}$ samples executions of the protocol until it obtains n successes and then sets the estimate as $\tilde{\varepsilon} = n/N$, where N is the number of trials.

Using a standard tail bound, it can be seen that, except with probability $2^{-\Omega(n)}$, it holds that $\tilde{\varepsilon} \in (\frac{\varepsilon}{2}, 2\varepsilon)$; throughout the rest of the proof, we assume that this is indeed the case.

Next, $\mathcal{S}$ will find a slot where one of the functions g_{k_0} or g_{k_1} is evaluated correctly with high-enough probability. For simplicity of notation we assume that the protocol has exactly $m = d$ slots, however the same argument holds also for $m > d$. For $i \in [d]$, let R_i be the randomness used by the rWI^x verifier in the i-th slot. For $i \in [d-1]$, let S_i be the random variable describing $\mathcal{S}$'s messages in the i-th slot as well as the rWI^x randomness R_{i+1} used in the next slot. Let $\mathcal{S}_d$ be the random variable describing $\mathcal{S}$'s messages in the d-th slot. Note that S_i is determined by R_i and the query q sent in the i-th slot.

We will show that $\mathcal{S}$ can sample $S_1, \ldots, S_{i-1}$ such that the residual circuit for the i-th slot can be used to extract one of the two keys k_0 or k_1. To do so, $\mathcal{S}$ repeatedly samples protocol executions $S_1, \ldots, S_d$ until it obtains n independent protocol executions that are non aborting. For each of these n executions and each slot $i \in [d]$, $\mathcal{S}$ creates a two circuits $\mathcal{C}_0, \mathcal{C}_1$. For $b \in \{0,1\}$ the circuit $\mathcal{C}_b$ contains messages $S_1, \ldots, S_{i-1}$ hard-coded, and on input query q, it emulates the interaction in the i-th slot using q and the randomness R_i contained in S_{i-1} and outputs the answer a_b among the two answers (a_0, a_1) returned by the residual circuit, or $\perp$ if the residual circuit aborts. Provided that $(\varepsilon/4)^{1/d} \geq 2^{-\sqrt{n}}$, $\mathcal{S}$ runs the extractor of $\mathcal{G}$, with approximation parameter $(\varepsilon/4)^{1/d}$ on every one of the $2dn$ circuits, If $(\varepsilon/4)^{1/d} < 2^{-\sqrt{n}}$, the simulator aborts, which has a negligible $2^{-\Omega(\sqrt{n})}$ statistical effect on the simulated distribution.

Let B be the event that the simulator does "gets stuck"; that is, it fails to extract a trapdoor string z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$ from any of the $2dn$ circuits. We claim that, that B

occurs with probability at most $2^{-\Omega(n)}$, and therefore, the validity of the simulation follows from the WI property of rsWIAOK. We now prove this. For an analysis of the running time of $\mathcal{S}$ see the proof of Lemma 6.5.

Assume towards contradiction that there exists a distinguisher D and a polynomial p such that for infirmly many pairs $(x, w) \in \mathcal{R}_{\mathcal{L}}$:

$$|\Pr[D((\mathsf{P}(w), \mathsf{V}^*)(x)) = 1] - \Pr[D(\mathcal{S}(x)) = 1]| \geq \frac{1}{p(n)} \ ,$$

where $|x| = n$. Since the only difference between the simulation and the real execution is the witness used in the final proof stage, it follows from the WI property of the rsWIAOK that:

$$|\Pr[D((\mathsf{P}(w), \mathsf{V}^*)(x)) = 1] - \Pr[D(\mathcal{S}(x)) = 1 \mid \neg B]| \leq \mathrm{negl}(n) \ .$$

Therefore B must occur with probability at least $1/p(n) - \mathrm{negl}(n) \geq 1/2p(n)$. Recall that with probability $1 - \varepsilon(n)$, V^* aborts the initial interaction. In this case $\mathcal{S}$ never gets stuck, therefore it must be that $\varepsilon(n) \geq \Pr[B] \geq 1/2p(n)$.

We next bound the probability that the event B occurrs. For $i \in [d]$, let G_i be the event that the verifier does not abort in the i-th slot, and let $\neg H_i$ be the event that $S_1, \ldots, S_{i-1}$ are such that the verifier does not abort with probability at least $(\varepsilon/2)^{1/d}$ over the choice of the query q in the i-th slot, i.e. $\Pr[G_i \mid S_1, \ldots, S_{i-1}] \geq (\varepsilon/2)^{1/d}$. We claim that conditioned on sampling a non-aborting execution $S_1, \ldots, S_d$, the event $\neg H_i$ occurs for some $i \in [d]$, with probability at least $\frac{1}{2}$ over the choice of $S_i | S_1, \ldots, S_{i-1}$. Indeed, by the residual probability Claim 2.1, it holds that $\Pr[G_1, \ldots, G_d, H_1, \ldots, H_d] \leq \left((\varepsilon/2)^{1/d}\right)^d = \varepsilon/2$, when setting $k = m = d$, and $\delta = (\varepsilon/2)^{1/d}$, and thus

$$\begin{aligned} \Pr[\exists i : \neg H_i \mid G_1, \ldots, G_d] = \\ 1 - \Pr[H_1, \ldots, H_d \mid G_1, \ldots, G_d] = \\ 1 - \frac{\Pr[G_1, \ldots, G_d, H_1, \ldots, H_d]}{\Pr[G_1, \ldots, G_d]} \geq \\ 1 - \frac{\varepsilon/2}{\varepsilon} \ . \end{aligned}$$

It follows that, except with probability 2^{-n}, $\mathcal{S}$ samples some $S_1, \ldots, S_{i-1}$ such that $\neg H_i$ holds. Let $\mathcal{C}_0, \mathcal{C}_1$ be the circuits $\mathcal{S}$ constructs from the sample $S_1, \ldots, S_{i-1}$. Let b be the bit committed in C. Since $\neg H_i$ holds, we have that $\mathcal{C}_b(q) \neq \perp$ with probability at least $(\varepsilon/2)^{1/d}$ over the query q. Next we use the soundness of the rWI^x proof to argue that $\mathcal{C}_b(q) \in \{\perp, g_{\mathsf{k}_b}(q)\}$ except with negligible probability. Assume towards contradiction that there exists a polynomial $r(n)$ such that for infinitely many pairs $(x, w) \in \mathcal{R}_{\mathcal{L}}$:

$$\Pr_q[\mathcal{C}_b(q) \notin \{\perp, g_{\mathsf{k}_b}(q)\}] \geq \frac{1}{r(n)} \ ,$$

and since $\mathcal{S}$ successfully samples $\mathcal{C}_b$ (except with probability 2^{-n}) by sampling $S_1, \ldots, S_{i-1}$ randomly at most n times we have that:

$$\Pr_{S_1, \ldots, S_{i-1}, q}[\mathcal{C}_b(q) \notin \{\perp, g_{\mathsf{k}_b}(q)\}] \geq \frac{1}{n \cdot r(n)} - 2^{-n} \ .$$

It follows that there is a way to fix the values of $S_1, \ldots, S_{i-2}$ and the value of S_{i-1} excluding the randomness R_i such that:

$$\Pr_{R_i, q}[\mathcal{C}_b(q) \notin \{\perp, g_{\mathsf{k}_b}(q)\}] \geq \frac{1}{n \cdot r(n)} - 2^{-n} \ .$$

Since the circuit $\mathcal{C}_b$ only uses the randomness R_i to emulate the interaction of V^* with the honest rWI^x verifier in the i-th slot, and since the only way $\mathcal{C}_b(q) \notin \{\perp, g_{\mathsf{k}_b}(q)\}$ is if V^* is proving a false statement, we get a contradiction the soundness of the rWI^x. Therefore, there exists some negligible function $\nu(n)$ such that $\mathcal{C}_b(q) \in \{\perp, g_{\mathsf{k}_b}(q)\}$ except with probability $\nu(n)$. Recall that $\mathcal{C}_b(q) \neq \perp$ with probability at least $(\varepsilon/2)^{1/d}$ and since we proved that $\varepsilon(n) \geq 1/2p(n)$, we have that $\nu(n) \leq \frac{(\varepsilon/2)^{1/d}}{4n^2}$ for large enough values of n. Therefore using the non-black-box learnability of $\mathcal{G}$, when $\mathcal{S}$ runs the extractor with approximation parameter $(\tilde{\varepsilon}/4)^{1/d} \leq (\varepsilon/2)^{1/d}$ and the circuit $\mathcal{C}_b$, it will manage to extract a string z such that $y_b = \varphi(\mathsf{z})$ with probability $1 - (2^{-\Omega(n)} + \nu^{\Omega(1)}) \cdot |\mathsf{V}^*|^{O(1)}$. This contradicts the fact that the event B occurs with probability at least $1/2p(n)$. This concludes the proof that simulation is valid. □

Proof sketch of Lemma 6.16. The proof closely follows the proof of Lemma 6.8 (showing concurrent zero-knowledge from verifiable robust unobfuscatable functions). We describe the main modifications to the simulation and its analysis required to show that Protocol 6.2 satisfies concurrent zero-knowledge. The verifiable robust unobfuscatable function f_{k} in Protocol 6.1 is replaced by a pair of functions $g_{\mathsf{k}_0}, g_{\mathsf{k}_1}$. The verification key vk is replaced by $(\mathsf{vk}_0, \mathsf{vk}_1, \mathsf{C}_0, \mathsf{C}_1, \mathsf{C})$ where $\mathsf{vk}_i = (h_i, y_i)$ for $i \in \{0, 1\}$. In Protocol 6.1, the prover verified an answer a to a query q using the algorithm $\mathsf{Ver}_{\mathcal{F}}(\mathsf{vk}, q, a)$. In Protocol 6.2 the prover verifies an answer (that now consists of a pair of answers) a_0, a_1, via an interactive rWI^x protocol. Therefore, each function evaluation slot in the Protocol 6.2 requires several rounds instead of two messages as in Protocol 6.1. The "trapdoor witness" in the proof stage of Protocol 6.1 is a string z such that $(\mathsf{vk}, \mathsf{z}) \in \mathcal{R}_{\mathcal{F}}$. In Protocol 6.2 the trapdoor witness is a string z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$. In Protocol 6.1 the commitments sent from the verifier to the prover are perfectly binding. In Protocol 6.2 the commitment is only statistically binding, however, with overwhelming probability over the first prover's message r, the commitment Com_r is perfectly binding.

We modify the procedures EXTRACT and EMULATE invoked by the simulator. Recall that in the simulation of Protocol 6.1, the procedure EMULATE, on parameters (T, m), returned a circuit $\mathcal{C}$ that gets as input a query q and simulates V^* (using the same logic as the function MAIN) starting from the state T and using q as the prover message in the first slot to start. If during this simulation, the first slot ends, $\mathcal{C}$ obtains V^*'s response a and outputs it. Otherwise, if V^* aborts or starts m additional slots, before the first slot terminates, $\mathcal{C}$ outputs $\perp$. (The randomness for $\mathcal{C}$ is sampled by the procedure EMULATE and hard-coded into $\mathcal{C}$.)

The new EMULATE procedure will output a pair of *randomized* circuits $\mathcal{C}_0, \mathcal{C}_1$. For $b \in \{0, 1\}$, the circuit $\mathcal{C}_b$ is an interactive circuit that gets as input a query q and randomness r. $\mathcal{C}$ simulates V^* (using the same logic as the function MAIN) starting from the state T using q as the prover message in the first slot, and using r as the randomness for the rWI^x verifier in that slot. If during this simulation, the first slot ends and the rWI^x proof of that slot passes verification, $\mathcal{C}$ obtains V^*'s response a_b and outputs it. Otherwise, if V^* aborts or starts m additional slots, before the first slot terminates, $\mathcal{C}$ outputs $\perp$. We think of all the randomness used to simulate V^*, except r and q, as sampled by the procedure EMULATE and hard-coded into $\mathcal{C}$.

The new EXTRACT procedure will obtain the pair of circuits $\mathcal{C}_0, \mathcal{C}_1$ by calling EMULATE, sample n independent sets of random coins $r_1, \ldots, r_n$ and for every $i \in [n]$, run the extractor of the verifiable robust unobfuscatable function with $\epsilon = \frac{1}{32}$ on both circuits $\mathcal{C}_0(\cdot\,; r_i)$ and $\mathcal{C}_1(\cdot\,; r_i)$. If one of the $2n$ executions of the extractor outputs a string z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$, EXTRACT outputs z, otherwise it outputs $\perp$.

It follows from the same argument as in the proof of Lemma 6.8 that if the simulated view

is distinguishable from the view of V^* in an interaction with the honest prover, then there must exist an execution of EXTRACT (either called directly from MAIN or called recursively from an execution of EMULATE) such that the circuits $\mathcal{C}_0, \mathcal{C}_1$ returned by the call to EMULATE, satisfy:

$$\Pr_{q,r}[\mathcal{C}_b(q;r) \neq \perp] \geq \frac{1}{16} \ ,$$

for any $b \in \{0,1\}$, however, with noticeable probability, all the executions of the extractor fail to extract a string z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$. Assume towards contradiction that such an execution exists.

Let $\mathsf{C}, \mathsf{C}_0, \mathsf{C}_1$ be the commitments sent by the verifier in the session that contains the first slot simulated by $\mathcal{C}_0$ and $\mathcal{C}_1$. Let b be the bit committed in C, and let k_b the key committed in C_b. Let r be random coins for $\mathcal{C}_b$, and let G_1 be the event that r is such that:

$$\Pr_{q}[\mathcal{C}_b(q;r) \neq \perp] \geq \frac{1}{32} \ .$$

By a standard averaging argument, it holds that $\Pr[G_1] \geq \frac{1}{32}$. For $\delta \in [0,1]$, let $G_2(\delta)$ be the event that:

$$\Pr_{q}[\mathcal{C}_b(q;r) \in \{\perp, g_{\mathsf{k}_b}(q)\}] \leq \delta \ .$$

Since the only way $\mathcal{C}_b(q) \notin \{\perp, g_{\mathsf{k}_b}(q)\}$ is for V^* to give a convincing proof for a false statement, it follows from the soundness of the rWI^x that for any polynomial p (and large enough n) $\Pr[\neg G_2(\frac{1}{p(n)})] \leq \frac{1}{p(n)}$. Overall we have that with some constant probability, r satisfies $G_1 \wedge G_2$ and therefore the circuit $\mathcal{C}_b(\cdot\,;r)$, $\left(\frac{1}{32}, \frac{1}{p(n)}, \mathcal{D}\right)$-approximates g_{k_b} (recall that $\mathcal{D}$ is the input sampler for $\mathcal{G}$). It follows that with overwhelming probability, for at least one of the random strings r_i sampled by EXTRACT, extraction from the circuit $\mathcal{C}_b(\cdot\,;r_i)$ succeeds with probability $1 - \left(\frac{|\mathsf{V}^*|}{p(n)}\right)^{O(1)}$. Since the above holds for any polynomial p, we have that EXTRACT fails to output a string z such that $y_0 = \varphi(\mathsf{z})$ or $y_1 = \varphi(\mathsf{z})$ only with negligible probability, leading to a contradiction.

This concludes the proof that simulation is valid. The analysis of the running time of $\mathcal{S}$ is similar to that in the proof of Lemma 6.8. □

6.5. A 3-Message Simultaneous Resettable WI Argument of Knowledge. In this section, we construct a three-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 5.1, our protocol will use the idea of turning a single witness statement into a two independent-witnesses statement as done in [24, 16].

LEMMA 6.17. *Protocol 6.3 is a resettable WI argument of knowledge.*

Since the protocol is a three-message protocol, we can apply the [4] transformation, where V derives its randomness by applying a PRF to the transcript; as a corollary, we get the following theorem:

COROLLARY 6.18. *Assuming trapdoor permutations, there exists a three-message simultaneously-resettable WI argument of knowledge (with non-black-box knowledge extraction).*

We now prove the lemma.

Protocol 6.3

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:

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

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

Fig. 6.3: An rWI three-message Argument of Knowledge (implying srWIAOK)

Proof of Lemma 6.17. 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 witnesses 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 6.1. We think of the hybrids as two symmetric 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 we will show that 0.6 is indistinguishable from 1.6.

Hybrid 0.1*:* This hybrid 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 $((\mathsf{k}_0, w_0), (\mathsf{k}_1, w_0))$; formally, the witness also includes the randomness for the commitments C_0 and C_1, but for notational brevity, we shall omit it. In Table 6.1, the witness used in part 0 of the rZAP is referred to as zapw_0, and the one corresponding to 1 in zapw_1.

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 6.1: The sequence of hybrids; the bit b corresponds to the bit commitment C; the gray cells indicate the difference from the previous hybrid.

Hybrid 0.2*:* This hybrid differs from the previous one only in the witness used in the 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 for the plaintext $0^{|\mathsf{k}_b|}$, instead of the plaintext 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 and the previous one, follows by the black-box indistinguishability of $\mathcal{F}$ (Definition 3.8). 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 computational 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 $p_0 - p_1 \geq \varepsilon$, for some noticeable $\varepsilon = \varepsilon(n)$. 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 $\frac{1+\varepsilon}{2}$. 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 $\frac{p_0}{2} + \frac{1-p_1}{2} \geq \frac{1}{2} + \frac{\varepsilon}{2}$. 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 $\varepsilon = \varepsilon(n)$, then with probability at least $\varepsilon/2$ over its first message, it holds with probability at least $\varepsilon/2$ 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 $\varepsilon/2$ 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 3.8) 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 two circuits for both options of b, and try to extract from both, just as was done in the proof of Claim 5.2.) This completes the proof of Lemma 6.17. □

7. From Resettably-Sound ZK back to Unobfuscatable Functions. Our constructions of resettable protocols are based on robust unobfuscatable functions. In the converse direction, we show how to transform any resettably-sound ZK argument (for NP) to a family of (not necessarily robust) unobfuscatable functions (as defined in [5]).

The high-level idea is to consider the family of functions defined by the resettably-sound ZK verifier's next message function where the key of the function contains the verifier's input: a hard-to-prove statement and random coins. Using resettable soundness we show that a black-box learner cannot produce an accepting protocol transcript. However, given a circuit that computes the function exactly, we show that the ZK simulator be used to produce an accepting protocol transcript.

CONSTRUCTION 7.1. *Let* $\mathsf{PRG} : \{0,1\}^n \to \{0,1\}^{2n}$ *ba a length-doubling pseudo-random generator and let* $\mathcal{L}$ *be the image set of* PRG*. Let* (P, V) *be a* d*-round resettably-sound argument for* $\mathcal{L}$ *where* V *uses* m *bits of randomness. We construct a family of functions* $\{f_{\mathsf{k}}\}_{\mathsf{k} \in \{0,1\}^{n+m+1}}$ *that implements the honest verifier strategy on some "hard to prove statement". More precisely, we interpret the key of the function as a triplet* $\mathsf{k} = (x, \mathsf{r}, b)$*, where* x *is an input to* PRG*,* r *is randomness for* V *and* b *is a bit. The function* f_{k} *is defined as follows:*

1. *Given a special input* "STATEMENT"*,* f_{k} *outputs* $y = \mathsf{PRG}(x)$*.*
2. *Given a partial protocol transcript consisting of prover messages* $\mathsf{ts} = (p_1, \ldots, p_i)$ *for* $i < d$*,* f_{k} *executes the honest verifier* V *with the random tape* r *and the statement "*$y \in \mathcal{L}$*" and feeds it with the messages* $(p_1, \ldots, p_i)$ *as the first* i *prover messages.* f_{k} *outputs* v_i*, the next verifier message generated by* V*.*
3. *Given a full transcript* $\mathsf{ts} = (p_1, \ldots, p_d)$*,* f_{k} *executes* V_{r} *as above. If* V_{r} *accepts,* f_{k} *outputs* b*; otherwise, it outputs* $\perp$*.*

We sketch why $\{f_{\mathsf{k}}\}_{\mathsf{k} \in \{0,1\}^n}$ is a family of unobfuscatable functions; indeed:

1. b **is black-box unlearnable:** no poly-size algorithm L that is given black-box access to f_{k} where k is sampled uniformly, can guess b with more than negligible advantage (over a random guess). Indeed, the bit b remains information theoretically hidden, unless L produces a convincing proof for the fact that $y \in \mathcal{L}$. Since L only interacts with f_{k} as a black-box, we can easily transform L into a resetting adversary P^* for the resettably-sound ZK argument, which convinces the resettable verifier V with roughly the same advantage that L has in learning b. By the pseudo-randomness guarantee of PRG, P^* does the same given a statement "$y \in \mathcal{L}$" for a random

$y \in \{0,1\}^{2n}$. Since such a statement is false with overwhelming probability, this contradicts the resettable-soundness of (P, V).

2. b **is non-Black-box learnable:** there exists a PPT algorithm E that, given any circuit C that computes the function f_{k}, can learn the value of b. E uses the code of C to construct an interactive verifier V' for the resettably-sound ZK protocol. E then runs the ZK simulator $\mathcal{S}$ with the code of V' and obtains a transcript ts. It follows from the correctness of the simulation that ts is, indeed, an accepting transcript. E runs C on the sequence of prover messages in ts and obtains the bit b.

Acknowledgements. We thank Ran Canetti for many enlightening discussions and comments. We thank Vipul Goyal for interesting discussions on constructing robust unobfuscatable functions. We also thank Vipul for pointing out the work of [40] and its implication to simultaneous resettability from one-way functions. We thank Rafael Pass for sharing with us the his result and ideas with Chung and Seth [19], obtaining resettable soundness from one-way functions; although the focus and techniques used in our work are quite different, their innovative work has been one of the main inspirations for this research. 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 [5]. We thank Leo Reyzin for valuable discussions and advice. Finally, we thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] BOAZ BARAK, *How to go beyond the black-box simulation barrier*, in FOCS, 2001, pp. 106–115.

[2] ———, *Constant-round coin-tossing with a man in the middle or realizing the shared random string model*, in FOCS, 2002, pp. 345–355.

[3] BOAZ BARAK AND ODED GOLDREICH, *Universal arguments and their applications*, SIAM J. Comput., 38 (2008), pp. 1661–1694.

[4] BOAZ BARAK, ODED GOLDREICH, SHAFI GOLDWASSER, AND YEHUDA LINDELL, *Resettably-sound zero-knowledge and its applications*, in FOCS, 2001, pp. 116–125.

[5] 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, 2001, pp. 1–18.

[6] BOAZ BARAK AND YEHUDA LINDELL, *Strict polynomial-time in simulation and extraction*, in STOC, 2002, pp. 484–493.

[7] BOAZ BARAK AND AMIT SAHAI, *How to play almost any mental game over the net - concurrent composition via super-polynomial simulation*, Electronic Colloquium on Computational Complexity (ECCC), (2005).

[8] MIHIR BELLARE AND ODED GOLDREICH, *On probabilistic versus deterministic provers in the definition of proofs of knowledge*, in Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman, 2011, pp. 114–123.

[9] NIR BITANSKY AND OMER PANETH, *From the impossibility of obfuscation to a new non-black-box simulation technique*, in FOCS, 2012, pp. 223–232.

[10] ———, *On the impossibility of approximate obfuscation and applications to resettable cryptography*, in STOC, 2013, pp. 241–250.

[11] MANUEL BLUM, *How to prove a theorem so no one else can claim it*, in Proceedings of the International Congress of Mathematicians, 1986, pp. 1444–1451.

[12] ZVIKA BRAKERSKI, CRAIG GENTRY, AND VINOD VAIKUNTANATHAN, *(leveled) fully homomorphic encryption without bootstrapping*, TOCT, 6 (2014), p. 13.

[13] RAN CANETTI, ODED GOLDREICH, SHAFI GOLDWASSER, AND SILVIO MICALI, *Resettable zero-knowledge (extended abstract)*, in STOC, 2000, pp. 235–244.

[14] RAN CANETTI, HUIJIA LIN, AND OMER PANETH, *Public-coin concurrent zero-knowledge in the global hash model*, in TCC, 2013, pp. 80–99.

[15] RAN CANETTI, HUIJIA LIN, AND RAFAEL PASS, *Adaptive hardness and composable security in the plain model from standard assumptions*, in FOCS, 2010, pp. 541–550.

[16] CHONGWON CHO, RAFAIL OSTROVSKY, ALESSANDRA SCAFURO, AND IVAN VISCONTI, *Simultaneously resettable arguments of knowledge*, in TCC, 2012, pp. 530–547.

[17] KAI-MIN CHUNG, RAFAIL OSTROVSKY, RAFAEL PASS, MUTHURAMAKRISHNAN VENKITASUBRAMANIAM, AND IVAN VISCONTI, *4-round resettably-sound zero knowledge*, in TCC, 2014, pp. 192–216.

[18] KAI-MIN CHUNG, RAFAIL OSTROVSKY, RAFAEL PASS, AND IVAN VISCONTI, *Simultaneous resettability from one-way functions*, in FOCS, 2013, pp. 60–69.

[19] KAI-MIN CHUNG, RAFAEL PASS, AND KARN SETH, *Non-black-box simulation from one-way functions and applications to resettable security*, in STOC, 2013, pp. 231–240.

[20] YI DENG, VIPUL GOYAL, AND AMIT SAHAI, *Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy*, in FOCS, 2009, pp. 251–260.

[21] CYNTHIA DWORK AND MONI NAOR, *Zaps and their applications*, SIAM J. Comput., 36 (2007), pp. 1513–1543.

[22] URIEL FEIGE, DROR LAPIDOT, AND ADI SHAMIR, *Multiple noninteractive zero knowledge proofs under general assumptions*, SIAM J. Comput., 29 (1999), pp. 1–28.

[23] URIEL FEIGE AND ADI SHAMIR, *Zero knowledge proofs of knowledge in two rounds*, in CRYPTO, 1989, pp. 526–544.

[24] ———, *Witness indistinguishable and witness hiding protocols*, in STOC, 1990, pp. 416–426.

[25] CRAIG GENTRY, *Fully homomorphic encryption using ideal lattices*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009, 2009, pp. 169–178.

[26] ODED GOLDREICH, *Foundations of Cryptography: Basic Tools*, Cambridge University Press, New York, NY, USA, 2000.

[27] ODED GOLDREICH AND ARIEL KAHAN, *How to construct constant-round zero-knowledge proof systems for np*, J. Cryptology, 9 (1996), pp. 167–190.

[28] ODED GOLDREICH AND HUGO KRAWCZYK, *On the composition of zero-knowledge proof systems*, SIAM J. Comput., 25 (1996), pp. 169–192.

[29] 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, New York, NY, USA, 1989,

ACM, pp. 25–32.
[30] ODED GOLDREICH, SILVIO MICALI, AND AVI WIGDERSON, *Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems*, J. ACM, 38 (1991), pp. 691–729.
[31] SHAFI GOLDWASSER, SILVIO MICALI, AND CHARLES RACKOFF, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.
[32] VIPUL GOYAL, *Non-black-box simulation in the fully concurrent setting*, in STOC, 2013, pp. 221–230.
[33] VIPUL GOYAL AND ABHISHEK JAIN, *On the round complexity of covert computation*, in STOC, 2010, pp. 191–200.
[34] VIPUL GOYAL AND HEMANTA K. MAJI, *Stateless cryptographic protocols*, in FOCS, 2011, pp. 678–687.
[35] VIPUL GOYAL AND AMIT SAHAI, *Resettably secure computation*, in EUROCRYPT, 2009, pp. 54–71.
[36] SATOSHI HADA, *Zero-knowledge and code obfuscation*, in Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings, 2000, pp. 443–457.
[37] JOE KILIAN AND EREZ PETRANK, *Concurrent and resettable zero-knowledge in poly-loalgorithm rounds*, in STOC, 2001, pp. 560–569.
[38] SILVIO MICALI AND LEONID REYZIN, *Min-round resettable zero-knowledge in the public-key model*, in EUROCRYPT, 2001, pp. 373–393.
[39] MONI NAOR, *Bit commitment using pseudorandomness*, J. Cryptology, 4 (1991), pp. 151–158.
[40] RAFAIL OSTROVSKY AND IVAN VISCONTI, *Simultaneous resettability from collision resistance*, Electronic Colloquium on Computational Complexity (ECCC), (2012).
[41] RAFAEL PASS, *Simulation in quasi-polynomial time, and its application to protocol composition*, in EUROCRYPT, 2003, pp. 160–176.
[42] RAFAEL PASS AND ALON ROSEN, *Bounded-concurrent secure two-party computation in a constant number of rounds*, in FOCS, 2003, pp. 404–413.
[43] ———, *Concurrent nonmalleable commitments*, SIAM J. Comput., 37 (2008), pp. 1891–1925.
[44] ———, *New and improved constructions of nonmalleable cryptographic protocols*, SIAM J. Comput., 38 (2008), pp. 702–752.
[45] RAFAEL PASS, ALON ROSEN, AND WEI-LUNG DUSTIN TSENG, *Public-coin parallal zero-knowledge for np*, 2011.
[46] RAFAEL PASS, WEI-LUNG DUSTIN TSENG, AND DOUGLAS WIKSTRÖM, *On the composition of public-coin zero-knowledge protocols*, SIAM J. Comput., 40 (2011), pp. 1529–1553.
[47] MANOJ PRABHAKARAN, ALON ROSEN, AND AMIT SAHAI, *Concurrent zero knowledge with logarithmic round-complexity*, in FOCS, 2002, pp. 366–375.
[48] RANSOM RICHARDSON AND JOE KILIAN, *On the concurrent composition of zero-knowledge proofs*, in EUROCRYPT, 1999, pp. 415–431.
[49] ANDREW CHI-CHIH YAO, *Theory and applications of trapdoor functions (extended abstract)*, in 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982, 1982, pp. 80–91.