COA-Secure Obfuscation and Applications

Ran Canetti[1]*, Suvradip Chakraborty[2]**, Dakshita Khurana[3]***, Nishant Kumar[3]***, Oxana Poburinnaya[4]†, and Manoj Prabhakaran[5]‡

[1] Boston University
[2] ETH Zürich
[3] UIUC
[4] Unaffiliated
[5] IIT Bombay

Abstract. We put forth a new paradigm for program obfuscation, where obfuscated programs are endowed with proofs of "well formedness." In addition to asserting existence of an underlying plaintext program with an attested structure, these proofs also prevent mauling attacks, whereby an adversary surreptitiously creates an obfuscated program based on secrets which are embedded in other obfuscated programs. We call this new guarantee *Chosen Obfuscation Attacks (COA) security*.
We show how to enhance a large class of obfuscation mechanisms to be COA-secure, assuming subexponentially secure IO for circuits and subexponentially secure one-way functions. To demonstrate the power of the new notion, we also use it to realize:

– A new form of software watermarking, which provides significantly broader protection than current schemes against counterfeits that pass a keyless, public verification process.
– *Completely CCA* encryption, which is a strengthening of completely non-malleable encryption.

1 Introduction

General-purpose program obfuscation (developed in [3,16,20,7,17,13] and many other works) holds great promise for enhancing the security of software: Software can be distributed and executed without fear of exposing sensitive design secrets or keys hidden in the code. Furthermore, when executing obfuscated software, all intermediate states are guaranteed to remain hidden, even when both the hardware and the software components of the underlying platform are adversarial.

* Supported by the DARPA SIEVE project, contract No. #HR00112020023.
** Work done while at IST Austria; supported in part by ERC grant 724307.
*** Supported in part by DARPA SIEVE project contract No. #HR00112020024, a gift from Visa Research, and a C3AI DTI award.
† Work done in part while at Boston University.
‡ Supported by a Ramanujan Fellowship and Joint Indo-Israel Project DST/INT/ISR/P-16/2017 of Dept. of Science and Technology, India.

However, ubiquitous use of program obfuscation might actually make the security of software worse in other respects: Verifying properties of an obfuscated program becomes harder - it is essentially reduced to black-box testing the program. This is highly unsatisfactory, especially in situations where the source of the program is untrusted. Indeed, the very property that makes obfuscation a boon for software creators - namely the ability to hide secrets both in the code and in the functionality - is a bane for users of the software, unless those users put complete trust in the creators.

Another concern is that the use of program obfuscation makes it harder to verify whether a given program depends on other programs in "illegitimate ways", where legitimacy (and lack thereof) relates to both structural and functional dependence between programs. For instance, obfuscation might facilitate *software plagiarism* by hiding the fact that program A runs some (potentially proprietary) program B as a subroutine, without publicly disclosing this fact. Furthermore, obfuscation might facilitate hiding the fact that program A is a *mauled* version of B - i.e. that A's functionality surrepetitiously depends on the functionality of B. The latter can be a concern even regardless of how B is implemented.[6]

We define and realize a new notion of program obfuscation that addresses the above concerns. However, before we present the new notion and some applications, we point out prior approaches. To the best of our knowledge, the only existing general notion of obfuscation that provides the ability to verify properties of obfuscated programs is Verifiable Indistinguishability Obfuscation [2]. However, that notion provides only limited hiding guarantees: indistinguishability of the obfuscated versions of two functionally equivalent programs is guaranteed only if there is a short witness of their equivalence. Furthermore, it provides no guarantees against adversaries that maul honestly generated programs.

Mauling attacks have been considered in the context of non-malleability of obfuscation. Thid has so far been studied only in the context of virtual black box obfuscation of point functions and related functionalities [11,24] and is thus susceptible to strong impossiblity results [4]. In particular, no generally viable notion of non-malleable obfuscation has been proposed.

Defending against program plagiarism has been studied in the context of software watermarking [4]. However, that line of work has concentrated on detecting illicit programs that very closely preserve the functionality of the watermarked program [12] (or else preseve some cryptographic use of it [18]), and does not address more general forms of plagiarism – e.g., generating a seemingly legiti-

[6] One might expect that existing notions of obfuscation, such as indistinguishability obfuscation (IO), already defend against such mauling attacks. However, this expectation fails for programs whose code includes random keys that affect the functionality. For instance, IO does not appear to rule out the possibility that an adversary, given an obfuscated version of a puncturable pseudorandom function with a random key k, manages to generate another obfuscated program that computes the same function but with key $(k + 1)$.

mate obfuscated program that illicitly uses a given program as a subroutine, or even changes some internal parameters while preserving the overall design.

1.1 Our Contributions

We first summarize our main contributions and then elaborate on each one.

- *Definitions:* We show how to enhance a number of existing notions of program obfuscation so as to provide a strong flavor of verifiability, combined with mitigating malleability attacks. We call such enhancement "security against chosen obfuscation attacks" (or, COA security). For sake of specifity, we also formulate a self-contained notion of COA obfuscation which will be used throughout this work.[7]
- *Constructions:* We construct COA obfuscation, assuming subexponentially secure iO and one-way functions. More generally, we show how to enhance (or, fortify) any one out of a class of measures of secure obfuscation to provide COA security.
- *Applications:* COA-secure obfuscation is directly applicable in situations where a user wishes to verify some properties regarding the structure and functionality of a given obfuscated program. In addition, we use COA-secure obfuscation to construct *software watermarking* mechanisms that provide a new and powerful notion of security. Finally, we use COA-secure obfuscation to construct a *completely CCA*-secure public-key encryption scheme, which is a new notion of security that natually strengthens *completely non-malleable* public-key encryption [14], which in turn augments non-malleablity to consider mauling attacks against both the ciphertext *and the public key.*

1.2 Defining COA Obfuscation

The first main contribution of this work is in developing a security notion that incorporates meaningful secrecy, verifiability, and non-malleability guarantees, while still being realizable by general-purpose obfuscation algorithms. Several challenges face such an endeavor.

First, to let the user verify various properties of the program, we would like to emulate the effect of having a one message proof attached to the obfuscated program. Since we do not have any cryptographic setup, such a proof cannot be zero-knowledge. Still, we would like to ensure that this proof does not reveal anything about the program that is to be hidden by the obfuscation. We thus formulate a notion of hiding that's intertwined with the functionality of the obfuscated program.

Developing an appropriate notion of non-malleability proves equally challenging. In particular, it appears hard to effectively capture mauling attacks on the

[7] We also define a somewhat weaker variant, which only guarantees verifiability without any non-malleability guarantees. We then realize this variant with a simpler construction than the one used to obtain COA security. See more details in [8].

functionality of programs without resorting to "simulation-based" formalisms (as done in [11,24]), which would in turn be subject to general impossibility akin to VBB-obfuscation [4]. Indeed, an indistinguishability-based notion that avoids the need for a simulator appears to be warranted in order to preserve broad applicability.

We get around this difficulty by extending the notion of CCA-secure commitments (namely, commitments that are secure against chosen commitment attacks [9]) to our setting. Indeed, our notion (which is, in turn, a natural extension of security against chosen ciphertext attacks for public-key encryption [27,28] to the setting of obfuscation) provides the stongest-known viable form of non-malleability for obfuscation schemes.

That is, we consider obfuscators O that take as input a program C along with a predicate ϕ that represents some attestation on the structure and functionality of C. Next, we augment the process of executing an obfuscated program with an initial step aimed at verifying that this program "corresponds to a plaintext program that satisfies ϕ." Here, however, we somewhat relax the traditional deterministic verification process, and instead allow for *randomized* verification that, given a purported obfuscaed program $\hat{C}$, outputs eiter a reject symbol, or else a fully functional program $\tilde{C}$. We then require that: (a) whenever $\phi(C)$ holds, $V(O(C,\phi),\phi) = \tilde{C}$, where $\tilde{C}$ is functionally equivalent to C, and (b) For all strings $\hat{C}$ and predicates ϕ, the event where $V(\hat{C},\phi) = \tilde{C}$ where $\tilde{C} \neq \perp$, and there is no program C that is functionally equivalent to $\tilde{C}$ and such that $\phi(C)$ holds, occurs only with negligible probability.

We stress that the above definition postulates a two-step randomized process for generating a functional obfuscated program: the first step is carried out by O, while the second is carried out by V. Furthermore, while obfuscating a legitimate program C (i.e. $\phi(C) = 1$) always results with a progam $\tilde{C} = V(O(C,\phi),\phi)$ that is functionally equivalent to C, an adversarially generated string $\hat{C}$ might result in a random variable $\tilde{C} = V(\hat{C},\phi)$ where different draws from $\tilde{C}$ are different programs with completely different functionalities. (Still $\phi(\tilde{C})$ holds almost always.)[8]

Finally, we would like to require that, for "sufficiently similar" programs C_0, C_1, polytime adversaries be unable to distinguish $O(C_0,\phi)$ from $O(C_1,\phi)$, *even when given access to a de-obfuscation oracle* $O^{-1}(\cdot,\phi)$. That is, we consider adversaries that are given a challenge program $C^* = O(C_b,\phi)$ for $b \leftarrow \{0,1\}$, along with access to an oracle $O^{-1}(\cdot)$ that operates as follows: If $\hat{C} = C^*$ or $V(\hat{C},\phi) = \perp$, then $O^{-1}(\hat{C}) = \perp$. Else, $O^{-1}(\hat{C}) = C$, where C is the lexicographically first program that's functionally equivalent to $\hat{C}$ and where $\phi(C)$ holds. The verification guarantee implies that (w.h.p.) such a program exists when $V(\hat{C},\phi) \neq \perp$.

It remains to determine what makes programs C_0, C_1 "sufficiently similar". Here we consider a number of variants, that correspond to existing notions of

[8] Our randomized verification step is borrowed from that of Non-Interactive Distributionally Indistinguishable (NIDI) arguments, as developed in [22]. Indeed, as there, it appears to be an essential relaxation that is crucial for realizability.

security for plain obfuscation. One natural option, corresponding to plain iO, considers any pair C_0, C_1 of equal-size, functionally equivalent programs.

Another option, which turns out to be relatively simple to work in the context of our applications, corresponds to a slight simplification of the notion of obfucation of probabilistic circuits, specifically $\mathsf{X\text{-}Ind\text{-}pIO}$ [10]. That is, we consider samplers that sample triples (C_0, C_1, z), where C_0 and C_1 are programs and z is some auxiliary information. A sampler Samp is $\mathsf{admissible\ for}$ ϕ if both C_0 and C_1 satisfy ϕ, and in addition any poly size adversary A, given z and oracle access to a program, can tell whether this program is C_0 or C_1 only with subexponentially small advantage over $1/2$, when $(C_0, C_1, z) \leftarrow \mathsf{Samp}$. An obfusator (O, V) is $\mathsf{COA\ Secure}$ with respect to predicate ϕ if any polytime adversary A', that's given $C^* = O(C_b, \phi)$, z, where $((C_0, C_1, z) \leftarrow \mathsf{Samp}$, as well as oracle access to $O^{-1}(\cdot)$, can guess b only with advantage that's polynomially related to that of A.

The intuiton for why COA security guarantees non-malleability is the same as in the case of CCA commitment and CCA encryption: An adversary that manages to "maul" its challenge progrom $\hat{C}$ into a program $\hat{C}'$ that passes verification and such that the preimages of the resulting $\tilde{C}$ and $\tilde{C}'$ are related in some non-trivial way, can readily use this ability to break COA security via applying $O^{-1}(\tilde{C}')$ to obtain the plaintext program that is related to the preimage of its challenge C^*. It is stressed that here the "non trivial relation" may include both structural and functional properties of the plaintext programs.

1.3 Applications of COA obfuscation

COA-secure obfuscation is clearly directly applicable in situations where a user wishes to verify some properties regarding the structure and functionality of a program that is otherwise obfuscated (hence "opaque"). We further demonstrate the power of this notion via two applications: First, we define and construct a new notion of program watermarking which, while being formally incomparble with existing notions, significantly pushes the envelope of what's obtainable in this context. Second, we define and construct a new notion of completely-CCA secure encryption, nametly encryption that remain secure even in the presence of an oracle that decrypts adversarially chosen ciphertexts *with respect to adversarially chosen public keys.* In both cases, our constructions rely on COA-secure obfuscation in a crucial way.

A new approach to software watermarking. Existing formulations of watermarking (e.g. [12,15]) concentrate on *preventing* the creation of "counterfeit" programs that are close in functionality to the watermarked program and yet remain unmarked. Instead, we propose a way to *publicly detect* any program that stands in some pre-determined relation with the watermarked program, and is unmarked (or carries a different mark than the watermarked program). The new notion is incomparable with the existing ones: On the one hand, the proposed notion does not rule out the possibility of creating "jail-broken programs", it only guarantees that these programs will be detectable. Still, the detection algorithm is fixed and keyless, hence detection is inherently public and universal. Furthermore, the new notion identifies a significantly larger class of "software piracy" attackes, namely those attacks where the "forbidden similarity" between the jail-broken program and the original program may be defined via some predetemined relation that considers both the *structure* and the *functionality* of the jail-broken program and the watermarked program.

More specifically, our proposed notion of watermarking, with respect to a family $\mathcal{C}$ of programs and a relation $R(\cdot,\cdot)$, postulates a marking algorithm M and a (randomized) verification algorithm V with the following properties. The watermarking party chooses a program C from the family $\mathcal{C}$, along with a mark m, and applies M to obtain a watermarked program $\hat{C}$ such that:
(a) $V(\hat{C}) = (\tilde{C}, m)$, where $\tilde{C}$ is functionally equivalent to C. (That is, $\hat{C}$ passes verification, bears the mark m, and results in a program that's functionally equivalent to the original.)
(b) Any adversarially generated program $\hat{C}'$ where $V(\hat{C}') = (\tilde{C}', m')$ and such that there exists a program C' that's functionally equivalent to $\tilde{C}'$ and such that $R(C, C')$ holds, must have $m' = m$ except for negligible probability. (That is, if $\hat{C}'$ passes verification and the resulting program $\tilde{C}$ has a functionally equivalent "plaintext" program C' that stands in the specified relation with C, then $\hat{C}'$ has to bear the mark m.)

We note that this notion is very general, and in particular prevents potential plagiarism where the plagiarized program has very different functionality than the watermarked one, and yet uses the watermarked program as a subroutine or otherwise incorporates it in its code.

We then use subexponential COA obfuscation to construct watermarking schemes for any function family $\mathcal{C}$ and relation R where:
(a) the description of a program $C \leftarrow \mathcal{C}$ is "one way" with respect to the functionality of the program (i.e. the description is uniquely determined, yet hard to effciently extract, given sufficiently many input-output pairs), and:
(b) R is such that whenever $R(C, C')$ holds, knowledge of C' enables breaking the one wayness of C. That is, there is an algorithm that computes the description of C, given only C' and oracle access to C. As a concrete example, we consider watermarking a PRF, where the relation R holds only if two PRF circuits use the same key; relying on a "key-injective" PRF, this can be extended to a relation R that holds whenever two PRFs agree on their outputs for some input.

Application to Completely CCA Encryption. We formulate a new notion of security for public key encryption, which we call completely CCA (CCCA) secure encryption. Our new notion of security provides a strong form of non-malleability for encryption schemes, and is a stronger variant of the notion of completely non-malleable encryption of Fischlin [14]. The latter notion of secure encryption scheme rules out non-malleability even when a man-in-the-middle adversary "mauls" an honest (public-key, ciphertext) pair to produce a new public key and a new ciphertext where the corresponding plaintext is related to the original plaintext (according to some relation $\mathtt{R}$).

Informally, our notion of completely CCA security strengthens *CCA security* (rather than plain non-malleability) by allowing the adversary to have access to a *strong decryption oracle.* The adversary can query this decryption oracle adaptively with different (possibly related) combinations of public keys and ciphertexts. The decryption oracle brute-force finds a message and randomness string corresponding to the queried public-key/ciphertext pair (and if no such pair exists, it it returns a $\bot$). The security requirement is that the adversary must not be able to break CPA-security for a challenge public-key ciphertext pair even given unbounded queries to this oracle (as long as it does not query the oracle on the challenge pair).

As shown by Fischlin [14], even his (weaker) notion of completely non-malleable encryption is *impossible* to realize with respect to black-box simulation in the plain model (i.e., without any trusted setup assumptions). In this work we show, surprisingly, that it is possible to construct completely CCA-2 secure encryption (and hence completely non-malleable encryption) in the plain model from COA obfuscation. While the reduction is BB, the bound is avoided by the fact that we need *sub-exponential* assumptions to construct COA obfuscation to begin with.

Let us provide more detail about the new notion of CCCA encryption. We first formulate a strong variant of completely non-malleable encryption in the spirit of CCA-secure commitments. In this variant, the adversary has access to a standard decryption oracle (with respect to adversarially generated ciphertext and the original secret key), and also has access to an oracle that, essentially, takes an adversarially generated public key pk and ciphertext c, and returns a plaintext m such that c could potentiallby the result of encrypting m with public key pk (and some random string). We note that this is a very strong primitive, that in particular implies both completely non-malleable encryption and CCA-secure commitment.

We then show that the Sahai-Waters CCA secure encryption [29], when the obfuscation algorithm is COA-secure (with respect to a predicate ϕ that attests for the correct structure of the obfuscated program), rather than plain iO, is completely CCA secure.

We provide two alternative proofs of security of this scheme. One proof follows a blueprint similar to that of [29], with one major difference: when the adversary $\mathcal{A}$ queries the decryption oracle with a public-key ciphertext pair $(\tilde{\mathsf{pk}}, c)$ then:

- If $\tilde{\mathsf{pk}}$ matches the challenge public key, then there is a direct reduction to the Sahai-Waters game.
- If $\tilde{\mathsf{pk}}$ is different from the challenge public key, then the reduction uses the COA deobfuscation oracle to decrypt the ciphertext c. In more details, the reduction first invokes $c\mathcal{O}.\mathsf{Ver}$ on $\tilde{\mathsf{pk}}$ to obtain some program $\tilde{P}$. The verifiability property of $c\mathcal{O}$ guarantees that if $\tilde{P} \neq \bot$, then there must be a program P' such that $\tilde{P} = \mathsf{iO}(P'; r)$. Since, we assume iO is injective, the reduction can then use the deobfuscation oracle $\mathcal{O}^{-1}$ on $\tilde{P}$ to recover P'. Now, from this plaintext program P' it is possible to extract "secret" PRF keys $(\mathsf{K}_1, \mathsf{K}_2)$ by reading the description of P' (this is possible since our COA fortification

is defined for circuits that have some PRF keys embedded). These secret keys can then be used to decrypt c.

The second proof directly uses the notion of COA-secure obfuscation: We formulate an admissible sampler where each sample consists of two instances of the encryption progeam in the public key, along with an auxiliary input that enables the adversary to obtain a challenge ciphertext c^* where embedded in c^* depends on whether one has access to the first or the second instance of the encryption program. We then argue that:
(a) when the encryption programs are given as oracles, it is infeasible to distinguish the two cases, thus the sampler is admissible. (Here we essentially use the fact that the underlying symmetric encryption scheme is CCA.) We conclude that a even a COA adversary, that has access to the COA-obfuscated version of one of the two copies of the encryption algorithm, along with the same auxiliary input a de-obfuscation oracle, is still unable to distinguish the two cases.
(b) On the other hand, A CCCA attack against the scheme can be simulated by a COA adversary that has access to a COA-secure obfuscation of one of the two instances of the encryption algorithm, to the same auxiliary input as before, and to a de-obfuscation oracle that's used to respond to the de-encryption quesries of the CCCA attacker. This means that such CCCA attack must fail.

1.4 Constructing COA obfuscation

Before sketching our construction for COA obfuscation, it will be helpful to set aside the non-malleability requirement, and consider only the simpler question of fortifying an obfuscation scheme to obtain verifiability. Let $\mathcal{O}$ be an obfuscator, and let ϕ be an efficiently computable predicate on programs. Recall that the verifiable version of $\mathcal{O}$ with respect to ϕ is a pair of algorithms $(v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$ such that the following holds:

- Correctness: For any program C such that $\phi(C)$ holds, if $\hat{C} \leftarrow v\mathcal{O}.\mathsf{Obf}(C)$ and $\widetilde{C} \leftarrow v\mathcal{O}.\mathsf{Verify}(\hat{C})$, then we have that $\widetilde{C}$ and C are functionally equivalent. That is, $v\mathcal{O}.\mathsf{Verify}$ "accepts" $\hat{C}$ as $\widetilde{C}$.
- Verifiability: Conversely, if $\widetilde{C}$ is the result of $v\mathcal{O}.\mathsf{Verify}(\hat{C})$ for some string $\hat{C}$, then, except with negligible probability, there exists a program C such that $\phi(C)$ holds and $\widetilde{C}$ is functionally equivalent to C, or more precisely, that $\widetilde{C}$ is in the image of $\mathcal{O}(C)$.
- Obfuscation: $v\mathcal{O}$ guarantees the same level of indistinguishability as $\mathcal{O}$. That is, there exists an efficient transformation from distinguishers between $v\mathcal{O}.\mathsf{Obf}(C_1)$ and $v\mathcal{O}.\mathsf{Obf}(C_2)$, to distinguishers between $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$, for any distribution over pairs of programs $C_1, C_2 \in F$ that satisfy ϕ.
(The formal definition considers distribution over (C_1, C_2, z) where z is auxiliary information that will be given to the distinguishers. Also, we shall permit the transformation to suffer a (possibly sub-exponential) quantitative loss in the level of indistinguishability.)

A useful interpretation of the pair of algorithms $(v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$ is that they *together implement* $\mathcal{O}$. An intriguing side-effect of this is that the honest $v\mathcal{O}$ obfuscator (who runs $v\mathcal{O}.\mathsf{Obf}$) does not necessarily know the final obfuscated program $\widetilde{C}$ generated by $v\mathcal{O}.\mathsf{Verify}$, though it knows that it is of the form $\mathcal{O}(C)$. On the other hand, if the obfuscator is malicious, it could control the outcome of $v\mathcal{O}.\mathsf{Verify}$ to be fixed, or alternately, ensure that different runs of $v\mathcal{O}.\mathsf{Verify}$ results in obfuscations of *different programs* (without violating the verifiability condition). These "relaxations" of $v\mathcal{O}$ may appear inconsequential, since an honest obfuscator is only interested in fixing the functionality, and on the other hand, a malicious obfuscator could have randomly chosen the program it wants to obfuscate. But as we see below, these relaxations are crucial to realizing verifiability fortification.

Exploiting NIDI for Verifiability. A natural approach to obtaining verifiability would be to attach some form of a non-interactive proof to the obfuscated program, proving that it was constructed as $\mathcal{O}(C)$ for some C such that $\phi(C)$ holds. Unfortunately, NIZK is impossible without a trusted setup. The next best option would be to use a non-interactive witness indistinguishable (NIWI) proof system, which can indeed be realized without a setup [5,19,6], under a variety of assumptions. Indeed, this was the approach taken in [2]. However, since a NIWI proof can hide a witness only if alternate witnesses are available, attaching a NIWI to $\mathcal{O}(C)$ directly does not suffice; instead, [2] cleverly combines three obfuscations with a NIWI which only proves that two of them correspond to functionally equivalent programs, both satisfying the predicate (this necessitates the verifiable-equivalence restriction). By evaluating the three programs and taking the majority, the user is assured that they are using a valid $\mathcal{O}(C)$. Also, the variable witnesses introduced suffices to prevent the NIWI from breaking the indistinguishability guarantee that iO provides for pairs of functionally equivalent circuits. Unfortunately, this approach is closely tied to the specific hiding guarantee of iO, limited to functionally equivalent circuits, and further adds a technical requirement that there should be a short witness to the equivalence.

The relaxations we build into $v\mathcal{O}$ enable an alternative. Specifically, instead of attaching a proof to (one or more) programs, $v\mathcal{O}.\mathsf{Verify}$ is allowed to sample a program and a proof. This enables us to use a proof system which lets the verifier sample a statement and a proof together, where the statement comes from a distribution determined by the prover. Such a proof system was recently introduced in [22], under the name of *Non-Interactive Distributionally Indistinguishable* (NIDI) arguments. The hiding property that NIDI offers is that as long as two statement distributions are indistinguishable from each other, then adding the proofs does not break the indistinguishability (by more than a sub-exponential factor). With a statement distribution corresponding to $\mathcal{O}(C)$, we can directly use the prover and verifier in a NIDI argument as $v\mathcal{O}.\mathsf{Obf}$ and $v\mathcal{O}.\mathsf{Verify}$.

COA Security Fortification. For defining COA-security fortification, we consider an *injective* obfuscator $\mathcal{O}$, where injectivity means that the obfuscator does not map two distinct programs to the same obfuscated program. While this property may naturally be present in obfuscators with perfect functionality preservation, it is easy to add this to an obfuscator (without affecting hiding properties) by simply attaching a perfectly binding commitment of a program to the obfuscation. The reason we shall consider the given obfuscator $\mathcal{O}$ to be injective is so that we will be able to unambiguously refer to a *de-obfuscation oracle*.

Given an injective obfuscator $\mathcal{O}$, a COA fortification of $\mathcal{O}$, denoted $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ is a pair of efficient algorithms satisfying the following properties:

– Correctness: This is similar to that in the case of verifiability fortification, except that there is no requirement for the obfuscated to satisfy any predicate. That is, for any program C, if $\hat{C} \leftarrow c\mathcal{O}.\mathsf{Obf}(C)$, then we have that $\widetilde{C} \leftarrow c\mathcal{O}.\mathsf{Ver}(\hat{C})$ such that $\widetilde{C}$ is functionally equivalent to C.
– Verifiability: Again, this property is similar to that in the case of verifiability fortification, except for the predicate. That is, if $\widetilde{C} \leftarrow c\mathcal{O}.\mathsf{Ver}(\hat{C})$ then, except with negligible probability, $\widetilde{C}$ is in the codomain of $\mathcal{O}(C)$.
– COA-secure obfuscation: COA-security is defined analogous to how CCA security is defined for encryption or (more appropriately) commitment. Consider an adversary who tries to guess $b \leftarrow \{1, 2\}$ from $\hat{C}$, where $\hat{C} \leftarrow c\mathcal{O}.\mathsf{Obf}(C_b)$, and (C_1, C_2) are a pair of programs. In a chosen obfuscation attack (COA), the adversary can create purported obfuscations $\hat{C}' \neq \hat{C}$ and have them de-obfuscated as $\mathcal{O}^{-1}(c\mathcal{O}.\mathsf{Ver}(\hat{C}'))$. Note that we require $\mathcal{O}$ to be injective so that $\mathcal{O}^{-1}$ is well-defined (it outputs $\perp$ if the input is not in the codomain of $\mathcal{O}$). Any advantage that the adversary has in guessing b with access to this deobfuscation oracle should translate to a distinguishing advantage between $\mathcal{O}(C_1)$ and $\mathcal{O}(C_2)$ (without a deobfuscation oracle).

COA-Security Fortification from Robust NIDI. Our COA-security fortification uses non-interactive CCA-secure commitments, as well as NIDI arguments. CCA-secure commitments were introduced by [9], and a non-interactive construction based on NIDI arguments was given in [22], which suffices for our purposes. However, for the NIDI arguments used in our construction, we need a stronger security guarantee than in [22], namely *robustness* – a term that we borrow from [9] where it was used in a similar sense. A robust NIDI w.r.t. an oracle $\mathbb{O}$ retains its indistinguishability preservation guarantee for distinguishers that have access to $\mathbb{O}$. We discuss the construction of robust NIDI soon after describing COA fortification.

Our COA-security fortification $c\mathcal{O}$ uses a non-interactive CCA-secure commitment scheme com and NIDI arguments for NP that are robust against the decommitment oracle for com. The obfuscation $c\mathcal{O}.\mathsf{Obf}(C)$ generates a robust NIDI proof $\widehat{C}$ for the language consisting of pairs $(\widetilde{C}, c)$ where $\widetilde{C} \leftarrow \mathcal{O}(C)$ and $c \leftarrow \mathsf{com}(C)$. Note that C is fixed, but only the distribution over $(\widetilde{C}, c)$ is determined by the NIDI prover (obfuscator). The verifier $c\mathcal{O}.\mathsf{Ver}(\widehat{C})$ runs the NIDI verifier to transform $\widehat{C}$ into a pair $(\widetilde{C}, c)$ (or rejects it), and then outputs $\widetilde{C}$. Note that the CCA-secure commitment c is simply discarded by the verifier. The role of this commitment is, in the proof of security, to allow running a COA adversary – which expects access to the oracle $\mathcal{O}^{-1} \circ c\mathcal{O}.\mathsf{Ver}$ – using the decommitment oracle for com.

Constructing Robust NIDI. Our construction of a robust NIDI follows the outline in [22], except that we instantiate all primitives with those that retain their security guarantees in the presence of the oracle $\mathbb{O}$. In what follows, we outline this construction.

In a nutshell, a NIDI consists of an iO-obfuscated program that obtains as input the first message of an appropriate two-message proof system (satisfying ZK with superpolynomial simulation), and outputs a statement sampled from the input distribution, together with a proof. In [22], it was shown that the resulting system hides the distribution from which statements are sampled. Our construction of robust NIDIs modifies this template by requiring the underlying iO and ZK proof to be secure in the presence of the oracle $\mathbb{O}$. For any oracle with a finite truth table, we achieve this by assuming subexponential security of the underlying primitives, eg., by setting the iO security parameter large enough such that iO becomes secure against adversaries that store the underlying truth table.

2 Preliminaries

We use $x \leftarrow S$ to denote uniform sampling of x from the set S. $[n]$ is used to denote the set $\{1, 2, \ldots n\}$. For $x, y \in \{0,1\}^n$, $x \circ y$ denotes the inner product of x, y, i.e. if $x = x[1 \ldots n], y = y[1 \ldots n]$, $x \circ y = \bigoplus_{i \in [n]} x_i \cdot y_i$. Functional equivalance of two circuits C_1, C_2 is denoted by $C_1 \equiv C_2$. We refer to a circuit class as $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$, where $\mathcal{C}_\kappa$ consists of a set of circuits. In addition, whenever we consider a circuit class, we assume that it has a corresponding efficient predicate to check membership in the class, i.e. for circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$, there is a corresponding efficient predicate $\phi_\mathcal{C}$ s.t. $\phi_\mathcal{C}(\kappa, C) = 1$ if $C \in \mathcal{C}_\kappa$ and 0 otherwise. For a distribution $\mathcal{D}$ on domain $\mathcal{X}$, $\mathsf{Supp}(\mathcal{D})$ denotes the support of $\mathcal{D}$ on $\mathcal{X}$. We define puncturable PRFs and key-injectivity for puncturable PRFs below:

Definition 1 (Puncturable PRF). *For sets $\{0,1\}^n$ and $\{0,1\}^m$, a puncturable PRF with key space $\mathcal{K}$ consists of a tuple of algorithms* $(\mathsf{PRF.Eval}, \mathsf{PRF.Puncture}, \mathsf{PRF.pEval})$ *that satisfy the following two conditions.*

- **Functionality preserving under puncturing.** *For every $x^* \in \{0,1\}^n$, every $x \in \{0,1\}^n \setminus \{x^*\}$, and all $K \in \mathcal{K}$, we have:* $\mathsf{PRF.Eval}(K, x) = \mathsf{PRF.pEval}(K\{x^*\}, x)$*, where* $K\{x^*\} \leftarrow \mathsf{PRF.Puncture}(K, x^*)$*.*
- **Pseudorandomness at punctured points.** *For every $x^* \in \{0,1\}^n$, every $x \in \{0,1\}^n \setminus \{x^*\}$, and any PPT adversary $\mathcal{A}$, it holds that*

$$\left| \Pr\left[\mathcal{A}(K\{x^*\}, \mathsf{PRF.Eval}(K, x^*)) = 1\right] - \Pr[\mathcal{A}(K\{x^*\}, U_k) = 1] \right| = \mathsf{negl}(\kappa),$$

where $K \leftarrow \mathcal{K}, K\{x^*\} \leftarrow$ $\mathsf{PRF.Puncture}(K, x^*)$, and U_k is the uniform distribution over $\{0,1\}^k$.

2.1 Non-Interactive Distributionally Indistinguishable (NIDI) Arguments.

In a NIDI argument [22] for an NP language $\mathcal{L}$, the prover algorithm $\mathcal{P}$ is given a distribution $\mathcal{D}$ for sampling member-witness pairs, and it generates a program π which can be used (by the verifier algorithm $\mathcal{V}$) to verifiably generate a member of the language $\mathcal{L}$. The hiding property of a NIDI is that if two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are such that the members they generate are indistinguishable from each other (when the witnesses are held back), then the program π generated by the NIDI prover remains similarly indistinguishable, upto a "gap" ϵ. We formally recall the definition of this primitive from [22] below.

Definition 2 (Non-Interactive Distributionally-Indistinguishable (NIDI) Arguments). *A pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$ is a non-interactive distributionally-indistinguishable (NIDI) argument for* NP *language $\mathcal{L}$ with associated relation $R_{\mathcal{L}}$ if there exist non-interactive algorithms $\mathcal{P}$ and $\mathcal{V}$ that satisfy:*

– **Completeness:** *For every $\mathsf{poly}(\kappa)$-sampleable distribution*[9] $\mathcal{D} = (\mathcal{X}, \mathcal{W})$ *over instance-witness pairs in $R_{\mathcal{L}}$ such that $\mathsf{Supp}(\mathcal{X}) \subseteq \mathcal{L}$,*

$$\pi \in \mathsf{Supp}\left(\mathcal{P}(1^\kappa, \mathcal{D})\right) \implies \mathcal{V}(1^\kappa, \pi) \in \mathsf{Supp}(\mathcal{X}).$$

– **Soundness:** *For every ensemble of polynomial-length strings $\{\pi_\kappa\}_\kappa$ there exists a negligible function μ such that*

$$\Pr_{x \leftarrow \mathcal{V}(1^\kappa, \pi_\kappa)}\Big[(x \neq \bot) \wedge (x \notin \mathcal{L})\Big] \leq \mu(\kappa).$$

– **ϵ-Gap Distributional Indistinguishability:** *There exists an efficient transformation T on distinguishers such that for every $\mathsf{poly}(\kappa)$-sampleable pair of distributions $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ and $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$ over instance-witness pairs in $R_{\mathcal{L}}$ where $\mathsf{Supp}(\mathcal{X}_0) \cup \mathsf{Supp}(\mathcal{X}_1) \subseteq \mathcal{L}$, and every distinguisher D with*

$$\Big|\Pr[D(\mathcal{P}(1^\kappa, \mathcal{D}_0)) = 1] - \Pr[D(\mathcal{P}(1^\kappa, \mathcal{D}_1)) = 1]\Big| = \nu(\kappa)$$

the distinguisher $D' = T(D)$ satisfies:

$$\Big|\Pr[D'(\mathcal{X}_0) = 1] - \Pr[D'(\mathcal{X}_1) = 1]\Big| \geq \epsilon(\kappa) \cdot \nu(\kappa).$$

We have the following theorem from [22].

Theorem 1. *Assuming the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation, there exist* NIDI *arguments satisfying ϵ-gap distributional indistinguishability, for every $\epsilon(\kappa) = 2^{-o(\log^c \kappa)}$, for a constant $c > 1$.*

[9] Here, we slightly abuse notation and use $\mathcal{D}$ to also denote a circuit that on input uniform randomness, outputs a sample from the distribution $\mathcal{D}$.

2.2 CCA Commitments

A chosen-commitment attack (CCA) secure commitment scheme [9] is a commitment scheme, which remains hiding for commitments even in the presence of a (computationally inefficient) "decommitment oracle" $\textsc{cca}.\mathsf{DeCom}$ that opens all commitments that do not match the challenge commitment. For the decommitment oracle to be well-defined, we shall require that the commitment is perfectly binding: for all r_0, r_1 and $m_0 \neq m_1$ we have that $\textsc{cca}.\mathsf{Com}(m_0; r_0) \neq \textsc{cca}.\mathsf{Com}(m_1; r_1)$.

A CCA secure commitment scheme is parameterized by a message length $M = M(\kappa)$; we shall consider the message space to be $\{0,1\}^M$, where M is polynomial. As defined below, a *non-interactive* CCA commitment scheme consists of an efficient randomized algorithm $\textsc{cca}.\mathsf{Com}$ (with an implicit "canonical opening"). We let $\textsc{cca}.\mathsf{DeCom}$ denote the function that maps an output of $\textsc{cca}.\mathsf{Com}$ to the message underlying it (or $\perp$ if no such message exists).

Definition 3. *An $\epsilon(\kappa)$-secure* non-interactive CCA commitment scheme *over a message space $\{0,1\}^{M(\kappa)}$ consists of a randomized algorithm $\textsc{cca}.\mathsf{Com}$ and a deterministic algorithm $\textsc{cca}.\mathsf{DeCom}$, satisfying the following.*

– **Correctness.** *For all $m \in \{0,1\}^M$ and $r \in \{0,1\}^*$ we have that*

$$\textsc{cca}.\mathsf{DeCom}(\textsc{cca}.\mathsf{Com}(1^\kappa, m; r)) = m.$$

(This implies perfect binding.)

– **Efficiency.** $\textsc{cca}.\mathsf{Com}$ *runs in time* $\mathsf{poly}(\kappa)$*, while* $\textsc{cca}.\mathsf{DeCom}$ *runs in time* $2^{O(\kappa)}$.

– $\epsilon(\kappa)$**-Security.** *For a message $m \in \{0,1\}^M$ and a distinguisher $\mathcal{D}$, let*

$$p^{\text{CCA}}_{\mathcal{D},m} = \Pr_{c \leftarrow \textsc{cca}.\mathsf{Com}(1^\kappa, m)}[\mathcal{D}^{\textsc{cca}.\mathsf{DeCom} \circ \mathsf{Filt}_c}(1^\kappa, c) = 1],$$

where Filt_c is the identity function on all inputs except c, on which it outputs $\perp$. Then, for all polynomials s there is a negligible function ν such that, for all $m_1, m_2 \in \{0,1\}^M$ and all distinguishers $\mathcal{D}$ of size at most $s(\kappa)$,

$$\left| p^{\text{CCA}}_{\mathcal{D},m_1} - p^{\text{CCA}}_{\mathcal{D},m_2} \right| \leq \epsilon(\kappa)\nu(\kappa).$$

We rely on a recent construction of non-interactive CCA commitments from [22].

Theorem 2 ([22]). *Assuming sub-exponentially secure indistinguishability obfuscation and either*

– *Sub-exponential (classical) hardness of DDH and sub-exponential quantum hardness of LWE (as used in [21]), or*
– *Sub-exponential time-lock puzzles based on the RSW assumption (as used in [26])*

there exist non-interactive CCA commitments satisfying Definition 3.

The assumptions in the aforementioned theorem can also be reduced by using time-lock puzzles based on iO and the existence of hard-to-parallelize languages.

2.3 Obfuscation

An *obfuscator* $\mathcal{O}$ is a randomized program that probabilistically maps a circuit from some family $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ to another functionally equivalent circuit. We shall require an obfuscator to satisfy the following correctness and efficiency properties (with probability 1):

Functionality Preservation. For all $\kappa \in \mathbb{N}$ and all $C \in \mathcal{C}_\kappa$, $\mathcal{O}(1^\kappa, C) \equiv C$ (where $\equiv$ indicates that the two circuits are functionally equivalent).

Polynomial Slowdown. There exists a polynomial p such that for all $\kappa \in \mathbb{N}$ and all $C \in \mathcal{C}_\kappa$, $|\mathcal{O}(1^\kappa, C)| \le p(|C|)$ (where $|\cdot|$ denotes the size of a circuit).

Efficient Obfuscation. $\mathcal{O}$ is a polynomial time algorithm. Generally, we shall also assume that the circuits in $\mathcal{C}_\kappa$ are of size at most polynomial in κ.

Security. For a sampler Samp and a distinguisher D, we define, for $b \in \{1,2\}$,

$$p_{\mathcal{O},D}^{\mathsf{Samp},b} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widetilde{C}\leftarrow\mathcal{O}(1^\kappa,C_b)}}\left[D(\widetilde{C},z)=1\right]$$
$$\text{and}\quad \mathsf{Adv}_{\mathcal{O},D}^{\mathsf{Samp}} := \left|p_{\mathcal{O},D}^{\mathsf{Samp},1} - p_{\mathcal{O},D}^{\mathsf{Samp},2}\right| \tag{1}$$

Then, an obfuscator $\mathcal{O}$ is said to be $(\mathcal{S},\mathcal{D})$ secure, if for all $\mathsf{Samp} \in \mathcal{S}$ and $D \in \mathcal{D}$, $\mathsf{Adv}_{\mathcal{O},\mathcal{D}}^{\mathsf{Samp}}$ is negligible. In particular, for *indistinguishability obfuscation* (iO), $\mathcal{S}$ is the class of samplers which output (C_1, C_2, z) where $C_1 \equiv C_2$ and $z = (C_1, C_2)$, and $\mathcal{D}$ consists of all PPT distinguishers.

Following [10],[10] below we define a class of samplers, called admissible samplers, that only requires that it is (very) hard for a PPT adversary to distinguish between oracle access to C_1 and to C_2. Here the distinguishing probability is required to be negligible even after amplifying by a factor of 2^κ, with κ being the number of bits of inputs for the circuits.

Definition 4 (Admissible Samplers). *For any adversary $\mathcal{A}$, and $b \in \{1,2\}$, let*

$$p_{\mathcal{A}}^{\mathsf{Samp},b,\kappa} := \Pr_{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)}\left[\mathcal{A}^{C_b}(z)=1\right]$$
$$\textit{and}\quad \mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} := \left|p_{\mathcal{A}}^{\mathsf{Samp},1,\kappa} - p_{\mathcal{A}}^{\mathsf{Samp},2,\kappa}\right|.$$

A sampler Samp *over $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ where all $C \in \mathcal{C}_\kappa$ take κ-bit inputs, is called* admissible *if there exists a negligible function μ s.t. for any non-uniform PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Samp},\kappa} \le \mu(\kappa)\cdot 2^{-\kappa}$, for all sufficiently large κ.*

[10] Admissible samplers are a special case of X-Ind sampler defined in [10], where it is parametrized by a function $X(\kappa) \le 2^\kappa$. The definition of admissible samplers corresponds to setting $X(\kappa) = 2^\kappa$ and restricting to (deterministic) circuits taking κ-bit inputs.

We shall refer to an obfuscation scheme with respect to such admissible samplers as a pIO scheme. As shown in [10], assuming the existence of sub-exponentially secure iO and sub-exponentially secure puncturable PRFs, pIO schemes exist for any polynomial sized circuit family, that is secure against a class $\mathcal{D}$ of sub-exponential time distinguishers.

Next we define injective obfuscators.

Definition 5 (Injective Obfuscator). *An obfuscator $\mathcal{O}$ for a circuit family $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ is said to be* injective *if* $\forall \kappa_1, \kappa_2, C_1, C_2$

$$\mathcal{O}(1^{\kappa_1}, C_1; r_1) = \mathcal{O}(1^{\kappa_2}, C_2; r_2) \neq \bot \Rightarrow C_1 = C_2.$$

We remark that it is easy to convert any obfuscator into an injective obfuscator (without affecting its hiding properties) simply by attaching a perfectly binding commitment of the circuit to its original obfuscation.

3 New definitions

We define COA-secure obfucation in Section 3.1, and Verifiability/COA fortification for obfuscations in Sections 3.2 and 3.3 respectively. Towards this, first, we will need the following definition of circuit samplers.

Definition 6 (ϕ-Satisfying Samplers). *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class and ϕ be a predicate. We say that a randomized algorithm Samp is a* ϕ-satisfying sampler over $\mathcal{C}$ *if, for all large enough κ, $\mathsf{Samp}(1^\kappa)$ outputs (C_1, C_2, z) such that, with probability 1, $C_1, C_2 \in \mathcal{C}_\kappa$ and, $\phi(C_1) = \phi(C_2) = 1$.*

3.1 COA-Secure Obfuscation

Definition 7 (Admissible ϕ-satisfying Samplers). *A sampler algorithm $\mathsf{Samp}(1^\kappa)$ is an* admissible ϕ-satisfying sampler over $\mathcal{C}$ *if it is both admissible (according to Definition 4) and ϕ-satisfying (according to Definition 6) over $\mathcal{C}$.*

Definition 8 (COA-Secure Obfuscation). *A COA-secure obfuscation for a circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ w.r.t. a predicate ϕ is a pair of PPT algorithms $(c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ defined as follows*[11]*:*

- $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi) \to \widehat{C}$. *This takes as input the security parameter κ, a circuit $C \in \mathcal{C}_\kappa$, a predicate ϕ, and outputs an encoding $\widehat{C}$.*
- $c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi) \to \{\widetilde{C} \cup \bot\}$. *This takes as input a string $\widehat{C}$, a predicate ϕ, and outputs either a circuit $\widetilde{C}$ or a reject symbol $\bot$.*

These algorithms satisfy the following correctness, verifiability and security properties.

[11] Both the algorithms $c\mathcal{O}.\mathsf{Obf}$ and $c\mathcal{O}.\mathsf{Ver}$ take as input a predicate. This is to capture the uniformity of the algorithms w.r.t. ϕ.

– **Perfect Correctness.** *For every $\kappa \in \mathbb{N}$ and circuit $C \in \mathcal{C}_\kappa$ s.t. $\phi(C) = 1$, if $\widetilde{C} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi), \phi)$, then $\widetilde{C} \equiv C$.*
– **Verifiability.** *For every ensemble of polynomial-length strings $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that:*

$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa,\Pi_\kappa,\phi)}\left[\widetilde{C} \neq \perp \wedge \left(\nexists C \in \mathcal{C}_\kappa : \phi(C) = 1 \wedge \widetilde{C} \equiv C\right)\right] = \nu(\kappa).$$

– **COA Security.** *Let $\mathbb{O}$ be an oracle defined as follows: $\mathbb{O}(\kappa, \widetilde{C})$ outputs the lexicographically first circuit $C \in \mathcal{C}_\kappa$ such that $\phi(C) = 1$ and C is functionally equivalent to $\widetilde{C}$.*
For any sampler algorithm Samp, *and an oracle distinguisher $\mathcal{D}$, for $b \in \{1, 2\}$, let*

$$q^{\mathsf{Samp},b,\kappa}_{c\mathcal{O},\mathcal{D}} := \Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widehat{C}\leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa,C_b,\phi)}}\left[\mathcal{D}^{\mathbb{O}(\kappa,\cdot)\,\circ\, c\mathcal{O}.\mathsf{Ver}(1^\kappa,\cdot,\phi)\,\circ\,\mathsf{Filt}_{\widehat{C}}}(1^\kappa, \widehat{C}, z) = 1\right],$$

$$\mathsf{COAAdv}^{\mathsf{Samp},\kappa}_{c\mathcal{O},\mathcal{D}} := \left|q^{\mathsf{Samp},1,\kappa}_{c\mathcal{O},\mathcal{D}} - q^{\mathsf{Samp},2,\kappa}_{c\mathcal{O},\mathcal{D}}\right|$$

where $\mathsf{Filt}_{\widehat{C}}$ *denotes a function that behaves as the identity function on all inputs except $\widehat{C}$, on which it outputs $\perp$.*
Then for every admissible ϕ-satisfying sampler Samp *(according to Definition 7) and any non-uniform PPT distinguisher $\mathcal{D}$, there exists a negligible function $\mu(\cdot)$, s.t.* $\mathsf{COAAdv}^{\mathsf{Samp},\kappa}_{c\mathcal{O},\mathcal{D}} = \mu(\kappa)$.

While the above definition of COA security is w.r.t. admissible samplers, we can also define COA security more generally as an add-on for obfuscation schemes $\mathcal{O}$ whose security could be w.r.t. other samplers. Before presenting this notion of fortifying any obfuscation scheme with COA security, we introduce a simpler (but already useful) notion of fortifying an obfuscation scheme by adding verifiability.

3.2 Verifiability Fortification for Obfuscation

Given an obfuscation scheme $\mathcal{O}$, we shall define its verifiability fortification w.r.t. a predicate ϕ as a pair of algorithms $(v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$. The verification algorithm guarantees that, given a string Π (purportedly generated by $v\mathcal{O}.\mathsf{Obf}$), if $\widetilde{C} \leftarrow v\mathcal{O}.\mathsf{Verify}(\Pi)$ and $\widetilde{C} \neq \perp$, then there exists a circuit C which satisfies the predicate ϕ s.t. $\widetilde{C} = \mathcal{O}(C; r)$ for some randomness r.

Definition 9 (Verifiability Fortification for Obfuscation). *Let $\mathcal{O}$ be an obfuscator for a circuit class $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ and ϕ be an efficiently computable predicate on circuits. An ϵ-gap* verifiability fortification *of $\mathcal{O}$* w.r.t. *ϕ, is a tuple of PPT algorithms $v\mathcal{O} = (v\mathcal{O}.\mathsf{Obf}, v\mathcal{O}.\mathsf{Verify})$ that satisfy the following:*

– **Correctness.** *For every $\kappa \in \mathbb{N}$ and every circuit $C \in \mathcal{C}_\kappa$, such that $\phi(C) = 1$,*

$$\Pr_{\widetilde{C}\leftarrow v\mathcal{O}.\mathsf{Verify}(1^\kappa,v\mathcal{O}.\mathsf{Obf}(1^\kappa,C,\phi),\phi)}[\widetilde{C} \equiv C] = 1.$$

– **Verifiability.** *For every ensemble of polynomial-length strings $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that:*

$$\Pr_{\widetilde{C}\leftarrow v\mathcal{O}.\mathsf{Verify}(1^\kappa,\Pi_\kappa,\phi)}\left[\widetilde{C}\neq\bot\wedge\left(\nexists(C\in\mathcal{C}_\kappa,r):\phi(C)=1\wedge\widetilde{C}=\mathcal{O}(C;r)\right)\right]=\nu(\kappa).$$

– **ϵ-Gap Indistinguishability of Obfuscated Circuits.** *There exists an efficient transformation $\mathcal{T}$ (on distinguisher circuits) such that for any ϕ-satisfying sampler* Samp *(Definition 6) over $\{\mathcal{C}_\kappa\}_\kappa$ and distinguisher $\mathcal{D}$,*

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})}\geq\epsilon(\kappa)\cdot\mathsf{Adv}^{\mathsf{Samp}}_{v\mathcal{O}.\mathsf{Obf},\mathcal{D}}$$

where $\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O}',\mathcal{D}'}$ *(for $(\mathcal{O}',\mathcal{D}')=(\mathcal{O},\mathcal{T}(\mathcal{D}))$ or $(v\mathcal{O}.\mathsf{Obf},\mathcal{D})$) is as defined in* (1).

3.3 COA Fortification for Obfuscation

We now define COA fortification $c\mathcal{O}$ for an obfuscation scheme $\mathcal{O}$ w.r.t. a predicate ϕ. Apart from the natural correctness property, we require that $c\mathcal{O}$ satisfies verifiability w.r.t. predicate ϕ just like verifiability fortification. In addition, we want $c\mathcal{O}$ to satisfy "gap COA security", which intuitively means that any distinguisher $\mathcal{D}$ that distiguishes between $c\mathcal{O}.\mathsf{Obf}(C_1)$ and $c\mathcal{O}.\mathsf{Obf}(C_2)$ *given access to a circuit deobfuscation oracle* can be converted to a distinguisher that distinguishes $\mathcal{O}(C_1)$ from $\mathcal{O}(C_2)$ without access to any oracle. In our construction, our transformation between distinguishers is not necessarily of polynomial size in the security parameter κ – therefore, in addition to ϵ as before, we parameterize the gap security in our definition by $T=T(\kappa)$ to capture the (in)efficiency of this transformation.

Definition 10 (COA Fortification for Injective Obfuscators). *Let $\mathcal{O}$ be an injective obfuscator for a circuit class $\mathcal{C}=\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ and ϕ be an efficiently computable predicate on circuits. A (T,ϵ)-gap* COA fortification *of $\mathcal{O}$* w.r.t. *ϕ is a pair of PPT algorithms $c\mathcal{O}=(c\mathcal{O}.\mathsf{Obf},c\mathcal{O}.\mathsf{Ver})$ as follows:*

– $c\mathcal{O}.\mathsf{Obf}(1^\kappa,C,\phi)\to\widehat{C}$. *This is a randomized algorithm that on input security parameter κ, a circuit $C\in\mathcal{C}_\kappa$, and a predicate ϕ, outputs an encoding $\widehat{C}$.*
– $c\mathcal{O}.\mathsf{Ver}(1^\kappa,\widehat{C},\phi)\to\{\widetilde{C}\cup\bot\}$. *This is a randomized algorithm that on input security parameter κ, a string $\widehat{C}$, and a predicate ϕ, outputs either a circuit $\widetilde{C}$ or a reject symbol $\bot$.*

These algorithms satisfy the following correctness and security properties.

– **Perfect Correctness.** *For every $\kappa\in\mathbb{N}$ and every circuit $C\in\mathcal{C}_\kappa$ such that $\phi(C)=1$,*

$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa,c\mathcal{O}.\mathsf{Obf}(1^\kappa,C,\phi),\phi)}[\exists\, r\ s.t.\ \widetilde{C}=\mathcal{O}(1^\kappa,C;r)]=1$$

– **Verifiability.** *For every ensemble of polynomial-length strings* $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$*, there exists a negligible function* $\nu(\cdot)$ *such that:*

$$\Pr_{\widetilde{C}\leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa,\Pi_\kappa,\phi)}\left[\widetilde{C}\neq\bot\wedge\left(\nexists(C\in\mathcal{C}_\kappa,r):\phi(C)=1\wedge\widetilde{C}=\mathcal{O}(1^\kappa,C;r)\right)\right]=\nu(\kappa).$$

– (T,ϵ)**-Gap Security.** *Let* $\mathcal{O}^{-1}(\widetilde{C})=\begin{cases}C & \text{if } \exists(C\in\mathcal{C}_\kappa,r) \text{ s.t. } \widetilde{C}=\mathcal{O}(1^\kappa,C;r)\\ \bot & \text{otherwise.}\end{cases}$

(well-defined since $\mathcal{O}$ *is injective). For any* ϕ*-satisfying sampler* Samp *(see Definition 6), and an oracle circuit* $\mathcal{D}$*, for* $b\in\{1,2\}$*, let*

$$q^{\mathsf{Samp},b}_{c\mathcal{O},\mathcal{D}}:=\Pr_{\substack{(C_1,C_2,z)\leftarrow\mathsf{Samp}(1^\kappa)\\ \widehat{C}\leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa,C_b,\phi)}}\left[\mathcal{D}^{\mathcal{O}^{-1}\circ c\mathcal{O}.\mathsf{Ver}(1^\kappa,\cdot,\phi)\circ\mathsf{Filt}_{\widehat{C}}}(1^\kappa,\widehat{C},z)=1\right]$$

$$\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}:=\left|q^{\mathsf{Samp},1}_{c\mathcal{O},\mathcal{D}}-q^{\mathsf{Samp},2}_{c\mathcal{O},\mathcal{D}}\right|$$

where $\mathsf{Filt}_{\widehat{C}}$ *denotes a function that behaves as the identity function on all inputs except* $\widehat{C}$*, on which it outputs* $\bot$*.*

Then, there exists a T*-sized transformation* $\mathcal{T}$ *(on distinguisher circuits) such that for any admissible sampler* Samp *over* $\{\mathcal{C}_\kappa\}_\kappa$ *and distinguisher* $\mathcal{D}$*,*

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})}\geq\epsilon(\kappa)\cdot\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$$

where $\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})}$ *is as defined in (1).*

Remark 1. One could consider a (possibly) stronger definition that allows the sampler Samp used in defining $\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$ to also make de-obfuscation queries. We note that for worst-case indistinguishability notions for $\mathcal{O}$ (like iO), this does not make any difference, as the (non-uniform) sampler can output the optimal pair of circuits.

Remark 2. We remark that for any $T=T(\kappa)\geq\mathsf{poly}(\kappa)$, any $\epsilon=\epsilon(\kappa)\leq\mathsf{negl}(\kappa)$, (T,ϵ)-gap COA fortification for any injective (T,ϵ)-secure pIO implies COA-secure obfuscation according to Definition 8. Here (T,ϵ)-security indicates that the advantage of any $\mathsf{poly}(T)$-sized adversary in the pIO security game is at most $\mathsf{negl}(\epsilon)$.

4 Robust NIDI

Robust NIDI arguments w.r.t. an oracle $\mathbb{O}$ are an extension of NIDI arguments (Definition 2), whereby the gap distributional indistinguishability requirement of NIDI is further strengthened to hold even if the distinguisher has access to the oracle $\mathbb{O}$. (The completeness and soundness guarantees remain unchanged.) In other words, any distinguisher $\mathcal{D}^{\mathbb{O}}$, distinguishing the proofs generated by prover $\mathcal{P}$ on input the distributions on instance-witness pairs - $\mathcal{D}_0=(\mathcal{X}_0,\mathcal{W}_0)$ or $\mathcal{D}_1=(\mathcal{X}_1,\mathcal{W}_1)$, can be converted to an efficient distinguisher $T(D)^{\mathbb{O}}$ which distinguishes the underlying instances $\mathcal{X}_0$ or $\mathcal{X}_1$ upto a "gap" ϵ. We formally define the same below.

Definition 11 (Robust NIDI Arguments). *Let $\mathcal{L}$ be an* NP *language with an associated relation $R_{\mathcal{L}}$, and $\mathbb{O}$ be an arbitrary oracle. A NIDI argument for $\mathcal{L}$, $(\mathcal{P}, \mathcal{V})$ is said to be* robust w.r.t. $\mathbb{O}$ *if it satisfies the following:*

– **ϵ-Gap Robust Distributional Indistinguishability:** *There exists an efficient transformation T on distinguishers such that for every* $\mathsf{poly}(\kappa)$*-sampleable pair of distributions $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ and $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$ over instance-witness pairs in $R_{\mathcal{L}}$ where $\mathsf{Supp}(\mathcal{X}_0) \cup \mathsf{Supp}(\mathcal{X}_1) \subseteq \mathcal{L}$, and every distinguisher D with*

$$\left| \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_0)) = 1] - \Pr[D^{\mathbb{O}}(\mathcal{P}(1^\kappa, \mathcal{D}_1)) = 1] \right| = \nu(\kappa)$$

the distinguisher $\widehat{D} = T(D)$ satisfies:

$$\left| \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_0) = 1] - \Pr[\widehat{D}^{\mathbb{O}}(\mathcal{X}_1) = 1] \right| \geq \epsilon(\kappa) \cdot \nu(\kappa).$$

We construct robust NIDI arguments for any finite[12] oracle $\mathbb{O} = \{\mathbb{O}_\kappa\}_{\kappa \in \mathbb{N}}$ by modifying the construction in [22] to ensure that all the underlying primitives remain secure in the presence of oracle $\mathbb{O}$. Our approach to achieve this is to rely on complexity leveraging, although it may be possible to leverage other axes of hardness in order to instantiate the underlying primitives with those that remain secure in the presence of $\mathbb{O}$. In the full version [8], we prove the following theorem.

Theorem 3. *Fix any finite oracle $\mathbb{O} = \{\mathbb{O}_\kappa\}_{\kappa \in \mathbb{N}}$. Assuming the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation, there exist robust* NIDI *arguments w.r.t. $\mathbb{O}$, satisfying ϵ-gap distributional indistinguishability, for every $\epsilon(\kappa) = 2^{-o(\log^c(\kappa))}$, for some constant $c > 1$, satisfying Definition 11.*

5 Constructing COA Secure Obfuscation

In this section, we prove the following theorem.

Theorem 4. *For any $(T(\kappa), \epsilon(\kappa))$, if there exist $\epsilon(\kappa)$-secure CCA commitments satisfying Definition 3 for which the decommitment oracle can be implemented in time $T(\kappa)$, and robust NIDIs satisfying $\epsilon(\kappa)$-gap distributional indistinguishability w.r.t. the decommitment oracle for the CCA commitments (see Definition 11), then there exists a $(T(\kappa), \epsilon(\kappa)/4)$-gap COA fortification for any injective obfuscation, satisfying Definition 10.*

Here we describe our construction, and defer its proof of security to our full version [8].

[12] By 'finite', we mean that there exists a constant $c > 1$ s.t. for large enough κ the oracle $\mathbb{O}_\kappa$ can be represented as a truth-table of size at most 2^{κ^c}.

Construction 1. We require the following primitives:

- Let ccacom denote an $\epsilon(\kappa)$-secure CCA commitment scheme according to Definition 3 and let $\mathbb{O}$ denote the (deterministic, inefficient) oracle that implements the $\textsc{cca}.\mathsf{DeCom}$ algorithm for ccacom. That is, on input a commitment string com, the oracle $\mathbb{O}$ outputs either a message $m \in \{0,1\}^*$ or $\perp$. Also, let $T = \mathsf{poly}(|m|, 2^\kappa)$ (where $|m|$ denotes the size of message space for ccacom).
- Let $\mathsf{r\text{-}NIDI}$ denote a robust NIDI w.r.t. oracle $\mathbb{O}$ for language $\mathcal{L}_\phi$, defined below.
- Let $\mathcal{O}$ denote the underlying obfuscator for our COA fortification. We will assume that this obfuscator is secure against $\mathsf{poly}(T)$-sized adversaries. This can be achieved by appropriately scaling the security parameter for $\mathcal{O}$, since $\mathcal{O}$ is assumed to be subexponentially secure.
- Define language $\mathcal{L}_\phi = \{\{O, c\} : \exists (C, r_1, r_2) : O = \mathcal{O}(C; r_1) \wedge c = \mathsf{ccacom}(C; r_2) \wedge \phi(C) = 1\}$

The algorithm $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C, \phi)$ does the following:

- Define distribution $\mathcal{D}_C(r_1||r_2) = \{\mathcal{O}(C; r_1), c = \mathsf{ccacom}(C; r_2)\}$ for uniformly sampled r_1, r_2.
- Output $\pi \leftarrow \mathsf{r\text{-}NIDI}.\mathcal{P}(1^\kappa, \mathcal{D}_C, \mathcal{L}_\phi)$ computed using uniform randomness r_c.

The algorithm $c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi)$ does the following:

- Sample randomness $r_\mathcal{R}$.
- Output $y \leftarrow \mathsf{r\text{-}NIDI}.\mathcal{V}(1^\kappa, \pi; r_R)$.

In particular, for $\epsilon(\kappa) = 2^{-o(\log^c(\kappa))}$ and some constant $c > 1$, there exist $\epsilon(\kappa)$-secure CCA commitments satisfying Definition 3 for which the decommitment oracle can be implemented in time $T(\kappa)$ where $T(\kappa) = 2^{\kappa^\delta}$ for some constant $\delta > 0$, and by Theorem 3 there exist robust NIDI arguments satisfying $\epsilon(\kappa)$ gap distributional indistinguishability w.r.t. the decommitment oracle for the CCA commitments. Then, the theorem above implies that there exist $(2^{\kappa^\delta}, 2^{-o(\log^c(\kappa))})$-gap COA fortification for any injective obfuscation.

Corollary 1. *Assuming the existence of sub-exponentially secure one-way functions and sub-exponentially secure indistinguishability obfuscation, there exists COA-secure obfuscation for all polynomial-sized circuits, satisfying Definition 8.*

Proof. (Sketch) By [10], assuming the existence of sub-exponentially secure iO and sub-exponentially secure puncturable PRFs, there exist subexponentially secure pIO schemes for any polynomial sized circuit family. That is, there exists a constant $\delta > 0$ such that for $T = 2^{\kappa^\delta}$, and every $\mathsf{poly}(T)$-sized distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathsf{Samp}}_{\mathsf{pIO},\mathcal{D}} = \mathsf{negl}(T)$ where Samp is an admissible sampler according to Definition 4. This scheme can be made injective (while retaining T-security) by attaching a perfectly binding commitment of the circuit to its original obfuscation.

Furthermore, for $\epsilon(\kappa) = 2^{-o(\log^c(\kappa))}$ and some constant $c > 1$, there exist $\epsilon(\kappa)$-secure CCA commitments satisfying Definition 3 for which the decommitment oracle can be implemented in time $T(\kappa)$, and by Theorem 3 there exist robust NIDI arguments satisfying $\epsilon(\kappa)$ gap distributional indistinguishability w.r.t. the decommitment oracle for the CCA commitments. Then, the theorem above implies that there exist $(2^{\kappa^\delta}, 2^{-o(\log^c(\kappa))})$-gap COA fortification for any injective obfuscation and in particular, for the injective pIO scheme described above. This results in a COA-secure obfuscation scheme, whose correctness and verifiability are immediate from those of the COA fortification. Furthermore, by definition of fortification, this means there is a T-sized transformation $\mathcal{T}$ on distinguishers such that for any admissible sampler Samp and distinguisher $\mathcal{D}$,

$$\mathsf{Adv}^{\mathsf{Samp}}_{\mathcal{O},\mathcal{T}(\mathcal{D})} \geq \epsilon(\kappa) \cdot \mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}}$$

This implies that for any $\mathcal{T}$-sized distinguisher $\mathcal{D}$, $\mathsf{COAAdv}^{\mathsf{Samp}}_{c\mathcal{O},\mathcal{D}} = \mathsf{negl}(\kappa)$. □

6 Keyless Verifiable Watermarking

In this section, we describe an application of COA obfuscation to building watermarking schemes. We present a generalized abstraction called *keyless verifiable* watermarking. As a consequence we obtain watermarking for useful functionalities like PRFs as a special case of this abstraction.

In the following, we define our notion of watermarking, which generalizes the one in the recent work of Kitagawa et. al. [23] to capture publicly markable and extractable watermarking schemes without setup.

Definition 12 (Keyless Verifiable Watermarking). *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class s.t. $\mathcal{C}_\kappa$ consists of circuits with input length $n(\kappa)$ and output length $m(\kappa)$. For a distribution family $\mathcal{D}_\mathcal{C}$ and a relation R over $\mathcal{C}$, a $(\mathcal{D}_\mathcal{C}, R)$-*unremovable keyless verifiable watermarking scheme with a message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ *consists of two PPT algorithms* $(\mathsf{Mark}, \mathsf{Verify})$ *as follows:*

- $\mathsf{Mark}(1^\kappa, C, m)$*:* Mark *is a randomized algorithm that takes as input a circuit $C \in \mathcal{C}_\kappa$, a message (or mark) $m \in \mathcal{M}_\kappa$ and outputs a (marked) circuit $\widehat{C}$.*
- $\mathsf{Verify}(1^\kappa, \widehat{C})$*:* Verify *is a randomized algorithm that takes as input a (purportedly marked) circuit $\widehat{C}$ and outputs a pair (C', m'), where C' is a circuit or $\perp$, and $m' \in \mathcal{M}_\kappa \cup \{\perp\}$.*

They should satisfy the following properties:

- **Correctness.** *There exists a negligible function μ s.t. for any circuit $C \in \mathcal{C}_\kappa$ and message $m \in \mathcal{M}_\kappa$ it holds that*

$$\Pr_{(C',m')\leftarrow\mathsf{Verify}(1^\kappa,\mathsf{Mark}(1^\kappa,C,m))}[C' \not\equiv C \ \vee \ m' \neq m] \leq \mu(\kappa).$$

- $(\mathcal{D}_\mathcal{C}, R)$**-Unremovability.** *There exists a negligible function ν s.t. for every non-uniform PPT adversary $\mathcal{A}$, for all sufficiently large κ,*

$$\Pr[\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa) = 1] \leq \nu(\kappa)$$

where the experiment $\mathsf{Exp}_{\mathcal{A},\mathcal{D}_\mathcal{C},R}(\kappa)$ is defined as follows:

1. *$\mathcal{A}(1^\kappa)$ sends a message $m \in \mathcal{M}_\kappa$ to the challenger. The challenger samples a circuit $C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}$ and responds with $\widehat{C} \leftarrow \mathsf{Mark}(1^\kappa, C, m)$.*
2. *$\mathcal{A}$ outputs a circuit $\widehat{C}^*$. Let $(C^*, m^*) \leftarrow \mathsf{Verify}(1^\kappa, \widehat{C}^*)$. Then, the experiment outputs 1 iff $C^* \neq \perp$, $m^* \neq m$, and*
   - *either $\exists C' \in \mathcal{C}_\kappa$ s.t. $C' \equiv C^*$ and $R_\kappa(C', C) = 1$,*
   - *or there is no circuit in $\mathcal{C}_\kappa$ that is functionally equivalent to C^*.*

Our definition is incomparable with recent related definitions, specifically those of Cohen et al. [12] Aaronson et al. [1], where the latter proposes a unified definition to capture most prior works. Specifically, we require that a watermarking scheme has a verification algorithm that is executed before running the watermarked programs. In our definition, the adversary is considered to have removed the watermark only if it produces a circuit that verifies, and for which the corresponding circuit in the circuit family is related to the original circuit.

Our definition also strengthens the definitions from prior works (including [23] and [1]) in some crucial ways:

– Our definition eliminates the need for any key generation algorithm/public parameters.
– Our definition incorporates a guarantee that a circuit passing the verification indeed belongs to the circuit class.

In addition, our definition has a flavor of traitor-tracing security that is similar to the recent works of [18]. In particular, we say that an adversary wins the watermarking game if it removes/modifies the watermark and outputs a circuit that is *related* to the original circuit – where *related* refers to satisfying one of a large class of relations.

We shall construct a $(\mathcal{D}_\mathcal{C}, R)$-unremovable keyless verifiable watermarking scheme, when circuits drawn from $\mathcal{D}_\mathcal{C}$ are unlearnable from oracle access, but the relation R is such that a circuit becomes learnable given a related circuit (as made precise in Theorem 5). We first describe our construction before stating its security guarantee.

Construction 2. Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be a circuit class s.t. $\mathcal{C}_\kappa$ consists of circuits that take inputs of length $n(\kappa)$ and produce outputs of length $m(\kappa)$, and $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ be a space of polynomially long messages. For any $\kappa \in \mathbb{N}$ and any $m \in \mathcal{M}_\kappa$, let $\mathcal{C}'_\kappa = \{C_m \mid C \in \mathcal{C}_\kappa, m \in \mathcal{M}_\kappa\}$, where

$$C_m(x) = \begin{cases} m||C(0) & \text{if } x = 0 \\ C(x) & \text{otherwise.} \end{cases}$$

Let circuit class $\mathcal{C}' = \{\mathcal{C}'_\kappa\}_{\kappa\in\mathbb{N}}$ be the marked circuit class and ϕ' be its membership predicate, i.e. $\phi'(C) = 1$ iff $C \in \mathcal{C}'_\kappa$ (ϕ' will internally use $\phi_\mathcal{C}$, the membership predicate of $\mathcal{C}$).

Let $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ be COA obfuscation for $\mathcal{C}'$, w.r.t. predicate ϕ' (according to Definition 8). Instantiate the watermarking scheme for $\mathcal{C}$ w.r.t. message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ and relation R as follows:

- $\underline{\mathsf{Mark}(1^\kappa, C, m)}$: Return $c\mathcal{O}.\mathsf{Obf}(1^\kappa, C_m, \phi')$, where C_m is defined using C as above.
- $\underline{\mathsf{Verify}(1^\kappa, \widehat{C})}$: Let $C' \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{C}, \phi')$. Parse $C'(0)$ as $m||y$, where $m \in \mathcal{M}_\kappa$ and $y \in \{0,1\}^{m(\kappa)}$. (If $C' = \bot$, or the parsing above fails, return $(\bot, \bot)$.) Construct a circuit C'' such that
$$C''(x) = \begin{cases} y & \text{if } x = 0 \\ C'(x) & \text{otherwise.} \end{cases}$$
Return (C'', m).

We provide the following theorem which captures the security of the above construction. We provide a proof of this in our full version [8].

Theorem 5. *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$, $\mathcal{D}_\mathcal{C} = \{\mathcal{D}_{\mathcal{C}_\kappa}\}_{\kappa\in\mathbb{N}}$ and $R = \{R_\kappa\}_{\kappa\in\mathbb{N}}$ be ensembles of polynomial (in κ) sized circuits, distributions over those circuits and relations over those circuits, as follows:*

- *$\mathcal{C}_\kappa = \{E_\kappa(f, \cdot) \mid f \in \{0,1\}^{h(\kappa)}\}$, where E_κ is a polynomial sized circuit implementing a function $E_\kappa : \{0,1\}^{h(\kappa)} \times \{0,1\}^{n(\kappa)} \to \{0,1\}^{m(\kappa)}$, with $n(\kappa) \leq \kappa^c$ for a constant $c < 1$.*
- *For any circuit family $\mathcal{A} = \{\mathcal{A}_\kappa\}_{\kappa\in\mathbb{N}}$ where $\mathcal{A}_\kappa$ is of size $\mathsf{poly}(2^{n(\kappa)})$,*
$$\Pr_{C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa},\, C' \leftarrow \mathcal{A}_\kappa^{C(\cdot)}} [\, C' \equiv C\,] \leq \mathsf{negl}(2^{n(\kappa)}).$$
- *There is a family of polynomial (in κ) sized circuits $\mathrm{Rec} = \{\mathrm{Rec}_\kappa\}_{\kappa\in\mathbb{N}}$ such that,*
$$\Pr_{C \leftarrow \mathcal{D}_{\mathcal{C}_\kappa}} \left[\exists C' \in \mathcal{C}_\kappa,\ R_\kappa(C, C') = 1 \wedge \mathrm{Rec}_\kappa^{C(\cdot)}(C') \neq C\right] \leq \mathsf{negl}(\kappa).$$

Then the watermarking scheme in construction 2 is a $(\mathcal{D}_\mathcal{C}, R)$-unremovable keyless verifiable watermarking scheme, (according to Definition 12) for circuit class $\mathcal{C}$ and message space $\mathcal{M}$.

Next, we provide the following corollary which captures PRF watermarking as special case of the above theorem.

Corollary 2. *Let $F = \{F_k(\cdot)\}_{k\in\mathcal{K}_\kappa, \kappa\in\mathbb{N}}$ be a PRF family with key-space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$, and seed, input, and output lengths as polynomials $h(\kappa), n(\kappa)$ and $m(\kappa)$ respectively, such that $n(\kappa) \leq \kappa^c$ for some $c < 1$. In addition, suppose the key distribution ensemble $\mathcal{D}_\mathcal{K}$ and relation ensemble R are as follows:*

- *F is a sub-exponentially secure PRF under key distribution $\mathcal{D}_\mathcal{K}$. That is, for any adversary of size $\mathsf{poly}(2^{n(\kappa)})$, the following holds: (where $\mathcal{F}(n,m)$ = set of all functions with input length n and output length m)*
$$\left| \Pr_{k \leftarrow \mathcal{D}_\mathcal{K},\, b \leftarrow \mathcal{A}^{F_k(\cdot)}(1^\kappa)} [b = 1] - \Pr_{H \leftarrow \mathcal{F}(n,m),\, b \leftarrow \mathcal{A}^{H}(1^\kappa)} [b = 1] \right| \leq \mathsf{negl}(2^{n(\kappa)})$$

– *There exists an algorithm* Rec *s.t.*

$$\Pr_{k \leftarrow \mathcal{D}_{\mathcal{K}}} \left[\exists k' \in \mathcal{K},\ R_\kappa(k', k) = 1 \wedge \mathrm{Rec}_\kappa^{F_k(\cdot)}(k') \neq k \right] = \mathsf{negl}(\kappa).$$

Then the watermarking scheme for F in construction 2 is a $(\mathcal{D}_{\mathcal{K}}, R)$-unremovable keyless verifiable watermarking scheme.

As a concrete instantiation of the above corollary, we consider the following relation over PRF keys: $R_\kappa(k, k') = 1$ iff $F_k(\cdot)$ agrees with $F_{k'}(\cdot)$ on at least one input. We will use a sub-exponentially secure PRF family F, which satisfies the following *key injectivity* property:

$$\Pr_{k \leftarrow \mathcal{D}_{\mathcal{K}_\kappa}} [\exists k' \in \mathcal{K},\ R_\kappa(k, k') = 1 \wedge k' \neq k] = \mathsf{negl}(\kappa).$$

where $\mathcal{D}_{\mathcal{K}_\kappa}$ denotes the key distribution for which the PRF security holds. Such PRFs can be constructed as in [12] under sub-exponential DDH and LWE assumptions. For such a PRF, $R(k, k') = 1$ iff $k = k'$ (for most k). Then, letting Rec be the identity function satisfies the condition on the relation R in the above corollary. Thus, instantiating Corollary 2 with $F_k(\cdot), \mathcal{D}_{\mathcal{K}}, R$ as defined above, we get a $(\mathcal{D}_{\mathcal{K}}, R)$-keyless verifiable watermarking scheme for F.

7 Completely CCA-secure Encryption

In this section, we introduce the notion of a completely CCA-secure public key encryption scheme. Our notion of completely CCA secure PKE is a generalization of the notion of completely non-malleable encryption put forward by [14]. The original definition of Fischlin [14] follows a simulation-based formulation. Later [30] gave a game-based formulation of completely non-malleable encryption and showed it to be equivalent to the original simulation-based definition of complete non-malleability. Our formulation of completely CCA-secure encryption also uses a game-based formulation.

Definition 13 (C-CCA-security). *An encryption scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is completely CCA secure if there exists a (potentially randomized) verification algorithm* KeyVerify *such that the following hold.*

– *Soundness of verification: For any string $\hat{pk}$ and message x, the probability that $\mathsf{KeyVerify}(\hat{pk})$ rejects and $\mathsf{Enc}(\hat{pk}, x) \neq \perp$ is negligible, i.e., there is a negligible function $\mu(\cdot)$ in the security parameter κ such that*

$$\Pr[(r_v, r_e) \leftarrow \{0,1\}^\kappa,\ \mathsf{KeyVerify}(\hat{pk}; r_v) = 0\ \wedge\ \mathsf{Enc}(\hat{pk}, x, r_e) \neq \perp)] < \mu(\kappa)$$

– *For every PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{c\text{-}cca}_{\mathcal{PKE},\mathcal{A},b}(\cdot)$ is upper bounded by $\mu(\kappa)$, where*

$$\mathsf{Adv}^{c\text{-}cca}_{\mathcal{PKE},\mathcal{A},b}(\cdot) = \Pr[\mathsf{Exp}^{c\text{-}cca}_{\mathcal{PKE},\mathcal{A}}(\kappa) = 1] - \frac{1}{2}$$

and $\mathsf{Exp}^{c\text{-}cca}_{\mathcal{PKE},\mathcal{A}}(\kappa)$ is defined via the following experiment involving $\mathcal{A}$ and a (potentially inefficient) challenger $\mathcal{C}$:

1. *The challenger $\mathcal{C}$ samples $r^* \xleftarrow{\$} \{0,1\}^\kappa$ and runs $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\kappa, r^*)$. It then returns pk^* to $\mathcal{A}$. It also samples a random bit $b \xleftarrow{\$} \{0,1\}$, computes the challenge ciphertext $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}^*, b, r)$ for a random r, and returns c^* to $\mathcal{A}$.*
2. *At any point in the game, the adversary can make (multiple) decryption queries to the challenger with respect to either the given public key or different (potentially mauled) public keys. In particular, $\mathcal{A}$ gets access to an oracle $\mathcal{D}(\cdot,\cdot)$. The oracle $\mathcal{D}$ takes as input either a ciphertext c_i, or else a pair $(\tilde{\mathsf{pk}}_i, c_i)$. In the first case, if $c_i = c^*$ then $\mathcal{D}$ returns $\perp$. Else $\mathcal{D}$ returns $\mathsf{Dec}(\mathsf{sk}^*, c_i)$. In the second case, $\mathcal{D}$ first chooses a random string r. Next, if $c_i = c^*$ and $\mathsf{pk}_i = pk^*$, or else $c_i = \perp$, or $\mathsf{KeyVerify}(\mathsf{pk}_i, r) = \perp$, then $\mathcal{D}$ returns $\perp$. Otherwise, $\mathcal{D}$ brute-force finds the set of message-randomness pairs (m, r) such that $\mathsf{Enc}(\mathsf{pk}_i, m; r) = c_i$. Finally, it returns a random message from this set, or $\perp$ if this set is empty.*
3. *When $\mathcal{A}$ outputs a guess b', return 1 if $b' = b$.*

7.1 C-CCAsecure PKE scheme in the Plain model

In this section we show how to construct a completely CCA2 secure PKE scheme in the plain model (i.e., without any set up assumption). It is known from the work of Fischlin [14] that, it is *impossible* to construct even completely non-malleable encryption schemes for general relations w.r.t. black-box simulation in the standard model. Later works [30,25] overcome this impossibility result by relying on the common random or reference string model. In this work, we show how to construct a completely CCA2 secure encryption scheme (which is stronger than complete non-malleability) in the *plain* model from COA fortification of *indistinguishability obfuscators* (iO) and one-way functions. The use of sub-exponential assumptions allow us to bypass the impossibility result of Fischlin [14]. We now present the details of our construction. The main ingredients required for our construction as follows:

Construction 3. Let $\epsilon > 0$ be an arbitrary small constant s.t. $\epsilon < \delta$ and:

- Let $F_1 : \{0,1\}^{2\kappa} \to \{0,1\}$ and $F_2 : \{0,1\}^{2\kappa+1} \to \{0,1\}^\kappa$ be two puncturable pseudo-random functions that for security parameter 1^k satisfy 2^{k^ϵ}- security against (non-uniform) adversaries.
- Let $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^{2\kappa}$ be a PRG that's 2^{k^ϵ}- secure against (non-uniform) adversaries.
- Let $\phi(C)$ be the predicate asserting that C is a circuit of the form of Figure 1 with F_1, F_2 and G as specified above.
- Let $\mathsf{iO} = (\mathsf{iO.Obf}, \mathsf{iO.Eval})$ be *sub-exponentially secure* injective indistinguishability obfuscation scheme that for security parameter 1^k satisfies 2^{k^ϵ}-security against (non-uniform) adversaries.
- Let $c\mathcal{O} = (c\mathcal{O}.\mathsf{Obf}, c\mathcal{O}.\mathsf{Ver})$ be a COA fortification of an underlying *injective* indistinguishability obfuscator (iO) for circuits with respect to predicate ϕ.

We construct our completely CCA-2 secure encryption scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

1. $\mathsf{KeyGen}(1^\kappa)$: The key generation algorithm does the following:
   - Sample puncturable PRF keys K_1 for F_1 and K_2 for F_2.
   - Generate program $P_{\mathsf{K}_1,\mathsf{K}_2}$ defined in Figure 1.
   - Compute $\widehat{P} \leftarrow c\mathcal{O}.\mathsf{Obf}(1^\kappa, P_{\mathsf{K}_1,\mathsf{K}_2}, \phi)$.
   - Output $pk = \widehat{P}$, $sk = (\mathsf{K}_1, \mathsf{K}_2)$.

Hardwired: Puncturable PRF Keys $\mathsf{K}_1, \mathsf{K}_2$.

Input: Message $m \in \{0,1\}$, randomness $r \in \{0,1\}^\kappa$.
(a) Let $t = \mathsf{G}(r)$
(b) Set $c_1 = t$, $c_2 = F_1(\mathsf{K}_1, t) \oplus m$, and $c_3 = F_2(\mathsf{K}_2, c_1|c_2)$.
(c) Output $c = (c_1, c_2, c_3)$.

Fig. 1. Program $P_{\mathsf{K}_1,\mathsf{K}_2}$.

2. $\mathsf{Enc}(pk, m \in \{0,1\})$: The encryption algorithm does the following:
   - Sample randomness $r \in \{0,1\}^\kappa$
   - Run the randomized verification algorithm $\widetilde{P} \leftarrow c\mathcal{O}.\mathsf{Ver}(1^\kappa, \widehat{P}, \phi)$.
   - If $\widetilde{P} \neq \perp$, run $\widetilde{P}(m; r)$ to obtain $c = (c_1, c_2, c_3)$.
3. $\mathsf{Dec}(pk, sk, c = (c_1, c_2, c_3))$: The decryption algorithm does the following:
   - Check if $c_3 \overset{?}{=} \mathsf{F}_2(\mathsf{K}_2, c_1|c_2)$. If the check fails, output $\perp$. Otherwise, it continues.
   - Output $m' = \mathsf{F}_1(\mathsf{K}_1, c_1) \oplus c_2$

In [8] we also show that Complete CCA security of the above scheme holds whehever the obfuscation scheme used is COA secure as in Definition 8. That is:

Theorem 6. *Assume that the obfuscation scheme $\mathcal{O}$ in the above scheme is COA secure with respect to predicate ϕ. Then the scheme is complete CCA secure as in Definition 13.*

References

1. Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In *Annual International Cryptology Conference*, pages 526–555. Springer, 2021.
2. Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. Verifiable functional encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 557–587, 2016. Full version at https://eprint.iacr.org/2016/629.pdf.

3. Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115, 2001.
4. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.
5. Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.
6. Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, pages 401–427, 2015.
7. Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 79–109. Springer, 2020.
8. Ran Canetti, Suvradip Chakraborty, Dakshita Khurana, Nishant Kumar, Oxana Poburinnaya, and Manoj Prabhakaran. COA-Secure obfuscation and applications. IACR Cryptol. ePrint Arch., 2022.
9. Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions. In *FOCS 2010*, pages 541–550, 2010.
10. Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 468–497. Springer, 2015.
11. Ran Canetti and Mayank Varia. Non-malleable obfuscation. In Omer Reingold, editor, *TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 73–90. Springer, 2009.
12. Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM J. Comput.*, 47(6):2157–2202, 2018.
13. Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 256–287. Springer, 2021.
14. Marc Fischlin. Completely non-malleable schemes. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 779–790. Springer, 2005.
15. Rachit Garg, Dakshita Khurana, George Lu, and Brent Waters. Black-box non-interactive non-malleable commitments. Cryptology ePrint Archive, Report 2020/1197, 2020. https://eprint.iacr.org/2020/1197.
16. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476, 2013.

17. Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021.
18. Rishab Goyal, Sam Kim, Brent Waters, and David J Wu. Beyond software watermarking: Traitor-tracing for pseudorandom functions. *IACR Cryptol. ePrint Arch.*, 2020:316, 2020.
19. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012.
20. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020. https://eprint.iacr.org/2020/1003.
21. Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In *CRYPTO 2019*, pages 552–582, 2019.
22. Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 186–215. Springer, 2021.
23. Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. *arXiv preprint arXiv:2010.11186*, 2020.
24. Ilan Komargodski and Eylon Yogev. Another step towards realizing random oracles: Non-malleable point obfuscation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018*, volume 10820 of *Lecture Notes in Computer Science*, pages 259–279. Springer, 2018.
25. Benoît Libert and Moti Yung. Efficient completely non-malleable public key encryption. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP*, volume 6198 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2010.
26. Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 576–587. IEEE Computer Society, 2017.
27. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, page 427–437, New York, NY, USA, 1990. Association for Computing Machinery.
28. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.
29. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.
30. Carmine Ventre and Ivan Visconti. Completely non-malleable encryption revisited. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2008.