Obfuscation-based Non-black-box Simulation and Four Message Concurrent Zero Knowledge for NP

Omkant Pandey[1,2,⋆,⋆⋆], Manoj Prabhakaran[1,⋆], and Amit Sahai[2,⋆⋆]

[1] University of Illinois at Urbana Champaign, {omkant,mmp}@uiuc.edu
[2] UCLA and Center for Encrypted Functionalities, sahai@cs.ucla.edu

Abstract. We show the following result: Assuming the existence of *public-coin differing-input obfuscation* (pc-diO) for the class of all polynomial time Turing machines, then there exists a four message, fully concurrent zero-knowledge proof system for all languages in **NP** with negligible soundness error. This result is constructive: given pc-diO, our reduction yields an explicit protocol along with an *explicit* simulator that is "straight line" and runs in strict polynomial time. The obfuscation security property is used only to prove soundness.
Public-coin differing-inputs obfuscation is a notion of obfuscation closely related to indistinguishability obfuscation. Most importantly for our result, pc-diO does not suffer from any known impossibility results: recent negative results on standard differing-inputs obfuscation do not apply to pc-diO. Furthermore, candidate constructions for pc-diO for the class of all polynomial-time Turing Machines are known.
Our reduction relies on a new non-black-box simulation technique which does not use the PCP theorem. We view the development of this new non-black-box simulation technique as the main contribution of our work. In addition to assuming pc-diO, our reduction also assumes (standard and polynomial time) cryptographic assumptions such as collision-resistant hash functions.

1 Introduction

Zero-knowledge and program obfuscation. Zero-knowledge proofs, introduced by Goldwasser, Micali and Rackoff [GMR85] are the classical example of

⋆ Research supported in part by the NSF Grant 1228856.
⋆⋆ Research supported in part from a DARPA/ONR PROCEED award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11- 1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

the *simulation paradigm*. They allow a *prover* to convince a *verifier* that a mathematical statement $x \in \mathbf{L}$ is true while giving *no additional knowledge* to the verifier. Prior to 2001, all known zero-knowledge simulators used the (cheating) verifier V^* as a *black-box* to produce their output (called the simulated view). Barak [Bar01] demonstrated how to take advantage of verifier's program to build more powerful *non-black-box* simulation techniques.

Constructing and analyzing non-black-box simulators can be a challenging task.The reason why taking advantage of verifier's code is difficult is because of the intriguing possibility of *program obfuscation*. Roughly speaking, program obfuscation is a method to transform a computer program (say described as a Boolean circuit) into a form that is executable but otherwise completely "unintelligible." In its strongest form, an obfuscated program leaks no information about the program beyond its "functionality" or the "input-output behavior". Therefore, access to the obfuscated program is no better than having black box access to it. This property, as formalized by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI$^+$01], is called the *virtual black box* (VBB) security. It was shown in [BGI$^+$01] that VBB-secure obfuscation is impossible in general. In hindsight, this negative result shows why non-black-box (NBB) simulation is possible, despite the possibility that program obfuscation could hide nearly every useful aspect of the verifier's code.

Zero-knowledge, in particular non-black-box simulation, is intimately connected to program obfuscation. This connection has been explicitly studied in the works of Hada [Had00], and Bitansky and Paneth [BP12b, BP12a, BP13a], and alluded to in several other works, e.g., [HT99, Bar01]). In this work, we explore this line of research further, particularly in light of recent work showing the first plausible constructions of general-purpose obfuscation schemes [GGH$^+$13]. In particular, for the first time, we show that program obfuscation can be useful for designing new non-black-box simulation strategies that yield constant-round concurrent zero knowledge protocols.

General-purpose obfuscation. In 2013, Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH$^+$13] presented the first candidate construction for general-purpose obfuscation. Several formalizations for obfuscation have been proposed as alternatives to the impossible-to-achieve notion of VBB obfuscation. A basic definition, called *indistinguishability obfuscation* (iO) [BGI$^+$01], roughly speaking, guarantees that if two (same-size) programs C_0, C_1 are functionally equivalent, then their obfuscations are computationally indistinguishable. A closely related notion is that of *differing input obfuscation* (diO) [BGI$^+$01] which, roughly speaking, guarantees that the obfuscations of C_0 and C_1 are computationally indistinguishable *provided that* it is hard to find an input x such that $C_0(x) \neq C_1(x)$. Unfortunately, recently evidence was shown [BP13b, GGHW14] that the notion of diO is impossible to achieve in general, due to the existence of problematic contrived auxiliary inputs. However, very recently, Ishai, Pandey, and Sahai [IPS15] formulated the notion of *public-coin differing-inputs obfuscation* (pc-diO) in which no auxiliary input is allowed except for the *random coins of the sampler.*

This modification avoids the negative results of [BP13b, GGHW14], and indeed all previous negative results on obfuscation using auxiliary input [GK05, GK13], as all previous negative results using auxiliary input critically relied on the possibility of a secret being embedded within the auxiliary input. Because in pc-diO the auxiliary input is only allowed to be public randomness, this possibility is eliminated (please see [IPS15] for further details). Furthermore, [IPS15], building on [ABG$^+$13, BCP14], present candidate constructions of pc-diO for the class of all polynomial-time Turing machines which can accept inputs of unbounded polynomial length.

Our results. In this work we show how to use program obfuscation to build a new non-black-box simulation strategy that works for fully *concurrent* zero-knowledge. More specifically, we show that:

– If public-coin differing-input obfuscation (pc-diO) exists for the class of all polynomial time Turing machines with unbounded inputs, then there exists a constant round, fully concurrent zero knowledge protocol for **NP** with negligible soundness error. The protocol has an *explicit* simulator;[3] the simulator is "straight line" and runs in strict polynomial time. The security of the obfuscation is used only prove the soundness of our protocol.
– We also show how to implement the core ideas of the above protocol in only *four* rounds. That is, our new protocol requires sending only four messages between the prover and the verifier.

Our protocol can be instantiated using the constructions of [BCP14, ABG$^+$13, IPS15] which obfuscates polynomial time Turing machines that can accept inputs of variable length (at most polynomial in the security parameter). We stress that we are able to obtain an *explicit* simulator for our protocol *irrespective of the computational assumptions* underlying the constructions of differing-inputs obfuscation. This is because we use the security—i.e., the public-coin differing-inputs security property—of obfuscation only in proving the soundness of our protocol. The simulator only depends on the correctness or the *functionality* of the obfuscated program, and hence can be described explicitly.

Other than pc-diO, our reduction only assumes standard (polynomial time hardness) assumptions, namely injective one-way functions and collision-resistant hash functions. Interestingly, our reduction does not *explicitly* depend on CS-proofs or universal-arguments [Kil92, Mic94, Kil95, BG02]; in particular, if we instantiate the constructions of [IPS15, ABG$^+$13] using the SNARKs of Bitansky et al. [BCCT13] based on bilinear maps (which do not rely on the PCP theorem), we obtain an instantiation of our protocol that also does not rely on the PCP theorem.

[3] In some protocols, specifically those based on knowledge-type assumptions [HT99], by virtue of the assumption that there exists an "extractor," it is only possible to obtain an *existential* result that a simulator exists; however, the actual program of the simulator is not explicitly given in the security proofs.

The round complexity of our final protocol also sheds new light on the *exact* (as opposed to asymptotic) round-complexity of concurrent zero-knowledge. Even in the simpler case of *stand alone* zero knowledge, the best known constructions require at least four rounds [FS89], and historically, concurrent zero-knowledge has always required more rounds than stand-alone zero-knowledge.[4] Our four-round protocol, for the first time, closes the gap between the best known upper bounds on round complexities of concurrent versus standalone zero-knowledge protocols (whose simulators can be explicitly described).

In retrospect, the fact that obfuscation actually *helps* non-black-box simulation can be perplexing. Indeed, in all prior works along this line [Had00, BP12b, BP13a], the core ideas for simulation are of *opposite* nature: it is the *inability* to obfuscate the "unobfuscatable functions" that helps the simulator. In our case, similar to [BP12a], it is the *ability* to obfuscate programs that allows polynomial time simulation. We believe that our method can be useful in other settings as well where non-black-box simulation seems essential such as constant-round leakage-resilient zero-knowledge [Pan14, GJS11] or CCA secure commitments in sub-logarithmic rounds [CLP10, GLP$^+$15].

Paper organization. We start by discussing how to use program obfuscation to avoid the use of universal arguments in Barak's protocol in Section 1.1. This results in a stand alone ZK protocol with a "straight line" simulator. In Section 4, we discuss why the simulator of this protocol fails in the concurrent setting, and present a (substantially) different constant-round protocol which is concurrent ZK along with main proof ideas. In Section 5, we present an overview of our four-round concurrent-ZK protocol. The full details can be found in the full version of this work [PPS15].

1.1 Technical Overview: Non-black-box Simulation via Program Obfuscation

Let us start by considering the simplest approach to zero-knowledge from (the possibility of) program obfuscation. For now, let us restrict ourselves to the case of *stand alone* zero-knowledge for **NP**-languages. Let $x \in \mathbf{L}$ be the statement and R be the witness-relation.

One simple approach is to have the verifier send an obfuscation of the following program $M_{x,s}$ which contains a secret string $s \in \{0,1\}^n$: $M_{x,s}(a) = s$ if and only $R(x,a) = 1$ and $M_{x,s}(a) = 0^n$ otherwise. Let $\widetilde{M}_{x,s}$ denote the iO-secure obfuscation of $M_{x,s}$. The real prover can recover s by using a witness w to x. Further, if x is false, $M_{x,s}$ is identical to $M_{x,0^n}$ and therefore must hide s,

[4] Barak's (bounded-concurrent ZK) protocol [Bar01] and recent construction of Chung, Lin, and Pass [CLP13b] require at least six rounds even after optimizations; the recent protocol of Gupta and Sahai [GS12b] requires five rounds and does not have an explicit simulator.

ensuring the soundness.[5] This gives us a two-message, *honest verifier* ZK proof. However, this idea does not help the simulation against malicious verifiers.

To fix this, let us try to use Barak's preamble (called $\mathsf{GenStat}$ [Bar01]) which has the following three rounds: first, the verifier sends a collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^n$, then the prover sends a commitment c to 0^n (using a perfectly binding scheme Com), and then the verifier sends a string $r \in \{0,1\}^n$. The transcript defines a "fake statement" $\lambda = \langle h, c, r\rangle$. A "fake witness" ω for the statement λ consists of a pair (Π, u) such that $c = \mathsf{Com}(h(\Pi)\ ;\ u)$ and Π is a program of length $\mathsf{poly}(n)$ which outputs the string r on input the string c (say, in $n^{\log\log n}$ steps). If h is a good collision-resistant hash function, then it was shown in [Bar01, BG02], no efficient prover P^* can output a satisfying witness ω to the statement λ (sampled in an interaction with the honest verifier). However, a simulator can commit to $h(V^*)$ (instead of 0^n) so that it will have a valid witness to the resulting transcript λ.

Coming back to our protocol, we use this idea as follows. We modify our first idea, and require the verifier to send a the obfuscation of a new program $M_{\lambda,s}$ (instead of $M_{x,s}$) where $\lambda = \langle h, c, r\rangle$ is the transcript of $\mathsf{GenStat}$. The new program $M_{\lambda,s}$ outputs s if and only if it receives a valid witness ω to the statement λ (as described earlier) and 0^n on all other inputs. To prove the statement x will be proven by proving the knowledge of either a witness w to x or the secret s (using an ordinary *witness-indistinguishable proof-of-knowledge* (WIPOK)). A simulator can "succeed" in the simulation as before: it commits to verifier's program in c to obtain (an indistinguishable statement) λ, then uses the fake witness ω (which it now has) to execute the program $\widetilde{M}_{\lambda,s}(\omega)$ and learn s and complete the WIPOK using s.

We now draw attention to some important points arising due to the use of λ in the obfuscation (instead of x). First, the length of the fake witness ω that the simulator has depends on the length of the program of V^*. Since the protocol needs to take into account V^* of *every* polynomial length, the obfuscated program $\widetilde{M}_{\lambda,s}$ must accept inputs ω of arbitrary, a-priori unknown, polynomial length. In other words, the obfuscated program $\widetilde{M}_{\lambda,s}$ must be a *Turing machine* which accepts inputs of arbitrary, a-priori unknown, (polynomial) length. Therefore, we will have to use program obfuscation for Turing machines.

Second, the statement $\lambda = \langle h, c, r\rangle$ is not a "false" statement since an all powerful prover can always find collisions in h and obtain a satisfying input to $M_{\lambda,s}$. The only guarantee we have is that if λ is sampled as above, then it would be *hard* for any efficient prover—even those with a valid witness to x—to find a satisfying input for $M_{\lambda,s}$. Therefore, unlike before (when x was used instead of λ), obfuscations $\widetilde{M}_{\lambda,s}$ and $\widetilde{M}_{\lambda,0^n}$ are not guaranteed to be indistinguishable if we use an iO-secure obfuscation; this is because the Turing machines $M_{\lambda,s}$ and $M_{\lambda,0^n}$ are not functionally equivalent. Therefore, we will have to use diO-secure obfuscation (since finding a differing input is still hard for these programs). The security of diO is a subtle issue and we discuss it shortly.

[5] By security of iO, $\widetilde{M}_{x,s} \overset{c}{\approx} \widetilde{M}_{x,0^n}$ and $\widetilde{M}_{x,0^n}$ has no information about s.

By putting these ideas together, we actually a get a standalone ZK protocol for **NP** (summarized below). The protocol needs to use some kind of reference to s other than the obfuscated program. This is done by using a $f(s)$ where f is a one-way function. This protocol has a "straight line" simulator. Further, unlike Barak's protocol, this protocol does not use universal arguments (and hence the PCP theorem).

Standalone Zero-Knowledge using Obfuscation. The protocol has three stages.

1. Stage-1 is the 3 round preamble $\mathsf{GenStat}$: V sends a CRHF h, P sends a commitment $c = \mathsf{Com}(0^n; u)$ and V sends a random $r \leftarrow \{0,1\}^n$.
2. In stage 2, V sends $(f, \widetilde{s}, \widetilde{M}_{\lambda,s})$ where f is a one-way function, $\widetilde{s} = f(s)$, and $\widetilde{M}_{\lambda,s}$ is the obfuscation of *Turing machine* $M_{\lambda,s}$ described earlier and $\lambda = \langle h, c, r\rangle$ is the transcript of stage-1. V also proves that $(f, \widetilde{s}, \widetilde{M}_{\lambda,s})$ are correctly constructed (using a standard ZK proof).
3. In stage-3 P provess, using a standard WIPOK, the knowledge of "either a witness w to x or secret s such that $\widetilde{s} = f(s)$."

Standalone ZK of this protocol can be proven by following Barak's simulator which commits to the code of V^* and therefore has an ω for simulated statement λ such that $\widetilde{M}_{\lambda,s}(\omega) = s$ within a polynomial number of steps; the simulator computes s and uses it in the WIPOK. The soundness of the protocol relies on the diO-security of obfuscation. Indeed, following [Bar01], for a properly sampled λ, it is hard to find ω such that $M_{\lambda,s}(\omega) \neq M_{\lambda,0^n}(\omega)$, and therefore it is hard to distinguish $\widetilde{M}_{\lambda,s}$ from $\widetilde{M}_{\lambda,0^n}$ by diO-security of obfuscation. Now, soundness is argued using three hybrid experiments: first use the simulator of the ZK protocol in stage 2, then replace $\widetilde{M}_{\lambda,s}$ from $\widetilde{M}_{\lambda,0^n}$, and finally extract s from the WIPOK in stage 3 and violate the hardness of one-way function f (since x is false, extraction must yield s).

In Section 4, we will discuss why the simulator of this protocol fails in the concurrent setting, and new ideas will be needed to obtain a concurrent ZK protocol. In particular, we will make use of the DGS-oracle idea [DGS09].

Security of diO and the issue of auxiliary information. Continuing from our discussion above, it is clear that for our approach to work obfuscations $\widetilde{M}_{\lambda,s}$ and $\widetilde{M}_{\lambda,0^n}$ should be indistinguishable to a cheating P^*. However, this is not all: in addition to one of these programs, P^* also has access to the statement λ, which is *auxiliary information* about the two programs. Therefore, our approach works if we have diO secure w.r.t. auxiliary information (distributed according to λ).

Recent implausibility results of Garg et al. [GGHW14] cast serious doubts about the existence of diO w.r.t. arbitrary auxiliary information. While their result does not rule out the possibility of diO w.r.t. specific distributions, we should be extra careful to not rely on auxiliary information which keeps some "secret" such as an obfuscated code [GGHW14].

In our approach, the distribution of λ does not have to keep any secrets. In the language of [IPS15], this is public-coin auxiliary information. We show that our approach indeed works by only assuming that the obfuscation is *public-coin differing-input* secure.

1.2 Related Work

Concurrent zero-knowledge. From early on, it was understood and explicitly proven in [FS90, GK96], that zero-knowledge is not preserved under parallel repetition where multiple sessions of the protocol run at the same time. The more complex notion of concurrent zero-knowledge (cZK) was introduced and achieved by Dwork, Naor, and Sahai [DNS98] (assuming "timing constraints" on the underlying network). A large body of research on cZK studied the round-complexity of black-box concurrent ZK with improving lower bounds on the same [KPR98, Ros00, CKPR03]. The state of art is the lower-bound is by Canetti, Kilian, Petrank, and Rosen [CKPR03] who prove that black-box cZK requires at least $O(\log n/\log\log n)$ rounds where n is the length of the statements being proven. Prabhakaran, Rosen, and Sahai [PRS02], building upon the prior works of Richardson and Kilian [RK99] and Kilian and Petrank [KP01], presented a cZK protocol for **NP** which has $\widetilde{O}(\log n)$ rounds, matching the lower bound of [CKPR03].

The central open question in this area is to construct a constant round cZK protocol for **NP** languages based on standard (or at least reasonable) assumptions. Barak [Bar01] showed that in the *bounded concurrent* setting where there is an a-priori upper bound on the number of sessions, there exists a constant round non-black-box cZK protocol for **NP**; the protocol is based on the existence of collision-resistant hash functions [Bar01] and uses universal arguments [Kil92, Mic94, Kil95, BG02]. The communication complexity of Barak's protocol depends on the a-priori bound on the sessions.

It has proven difficult to extend Barak's NBB techniques to the setting of fully concurrent ZK (i.e., to unbounded polynomially many sessions) in $o(\log n)$ rounds. Nevertheless, NBB techniques have enjoyed great success resulting in the construction of resettable protocols [BGGL01, DL07, DGS09, GM11], non-malleable protocols [Bar02, PR05b, PR05a], leakage-resilient ZK [Pan14], bounded-concurrent secure computation [PR03, Pas04], adaptive security [GS12a], and so on. Bitanksy and Paneth [BP12a] showed that it is possible to perform non-black-box simulation using oblivious transfer (instead of collision-resistant hash functions and universal arguments). This eventually led to the construction of resettablly-sound ZK under one-way functions [BP13a, CPS13, COPV13]. Goyal [Goy13] presents a non-black-box simulation technique in the fully concurrent setting and achieves the first public-coin cZK protocol in the plain model.[6]

[6] The protocol requires $\mathsf{poly}(n)$ rounds. Canetti et al. [CLP13a] obtain a similar result, albeit in the "global hash" model where a global hash function—which the simulator cannot program—is known to all parties.

An alternative approach to construct round-efficient zero-knowledge proofs is to use "knowledge assumptions" [Dam91, HT99, BP04]. The recent work of Gupta and Sahai [GS12b] shows that such assumptions also yield a constant round concurrent ZK protocol for **NP**. However, all known ZK protocols based on knowledge-type assumptions do not yield an *explicit* simulator. This is because the knowledge-type assumptions assume the existence of a special "extractor" machine (which is not explicitly known); this extractor is used by the simulator of ZK protocols and only provides an "existential" result.

Chung, Lin, and Pass [CLP13b] recently presented the first construction of a constant-round fully concurrent ZK protocol which has an explicit simulator. Their result is based on a new complexity-theoretic assumption, namely the existence of so called "strong **P**-certificates."

Another alternative proposed in the literature is to assume some kind of a setup such as timing constraints, (untrusted) public-key infrastructure, and so on [DNS98, DS98, CGGM00, Dam00, Gol02, PTV10, GJO+13] or switch to super-polynomial time simulation [Pas03, PV08]. We will not consider such models further in this work.

Program obfuscation. After the strong impossibility results of [BGI+01], research in program obfuscation proceeded in two main directions. The first line of research focussed on constructing obfuscation for specific functionalities such as point functions and their variants, proxy re-encryption, encrypted signatures, hyperplanes, conjunctions, and so on[Wee05, LPS04, HRSV07, Had10, CRV10, BR13a]. The other line of research focussed on finding weaker definitions and alternative models. Goldwasser and Rothblum [GR07] considered the notion of *best possible obfuscation* (and is equivalent to iO when the obfusactor is polynomial time); and Bitansky and Canetti [BC10] considered *virtual grey box* security. Alternative models for obfuscation such as the hardware model were considered in [GIS+10, BCG+11].

After [GGH+13], an improved construction of iO was presented by Barak et. al. [BGK+13]. Further, in an idealized "generic encodings" model it is shown that VBB-obfuscation for all circuits can be achieved [CV13, BR13b, BGK+13]. These results often involve a "bootstrapping step"; Applebaum [App13] presents an improved technique for bootstrapping obfuscation. Further complexity-theoretic results appear in recent works of Moran and Rosen [MR13], and Barak et. al. [BBC+14].

Sahai and Waters [SW13] show that indistinguishability obfuscation is a powerful tool and use it to successfully construct several (old and new) cryptographic primitives; further applications of iO appear in [HSW13, BZ13, BCP14, KRW13, MO13]

Differing input obfuscation was studied by Ananth et. al. [ABG+13], who present a candidate construction of diO for the class of polynomial time Turing machines and demonstrate new applications. Another variant of their construction allows the Turing machines to accept variable length inputs. Concurrent work of Boyle, Chung, and Pass [BCP14] introduces a related notion of *extractability obfuscation* and shows this notion (and diO) are implied by iO when the programs differ only on polynomially many inputs. In addition, it also presents obfuscation for the class of polynomial time Turing machines, building upon the work of Brakerski and Rothblum [BR13a].Very recently, in concurrent and independent works, construction for bounded-space RAM programs were presented by relying on iO and OWFs [BGT14] and other additional assumptions [CHJV14, LP14].

The issue of *auxiliary information* in program obfuscation was first considered by Goldwasser and Kalai [GK05], and further explored in [GK13, BCPR13, BP13b, GGHW14, IPS15]. The work of Bitansky, Canetti, Paneth, and Rosen [BCPR13] shows that if iO exists then "extractability primitives" such as knowledge-types assumptions and extractable one-way functions [CD09] cannot exist in the presence of *arbitrary* auxiliary information. Boyle and Pass [BP13b] strengthen this result further by showing a pair of (universal) distributions $\mathcal{Z}$,$\mathcal{Z}$' on auxiliary information such that either extractable OWF w.r.t. $\mathcal{Z}$ do not exist or extractability-obfuscations w.r.t. $\mathcal{Z}$' do not exist. The work of Garg, Gentry, Halevi, and Wichs [GGHW14] shows that diO w.r.t. arbitrary auxiliary information cannot exist if certain specific obfuscation assumption is true. Ishai, Pandey, and Sahai [IPS15] formulate the notion of public-coin diO in which the auxiliary is restricted to be merely the random coins of the sampler, and recover much of the existing applications of diO under this new notion.

2 Preliminaries

We use standard notations which are recalled here. This section can be skipped without affecting readability.

Notation. For a randomized algorithm A we write $A(x;r)$ the process of evaluating A on input x with random coins r. We write $A(x)$ the process of sampling a uniform r and then evaluating $A(x;r)$. We define $A(x,y;r)$ and $A(x,y)$ analogously. We denote by $\mathbb{N}$ and $\mathbb{R}$ the set of natural and real numbers respectively. The concatenation of two string a and b is denoted by $a \parallel b$.

We assume familiarity with interactive Turing machines (ITMs). For two randomized ITMs A and B, we denote by $[A(x,y) \leftrightarrow B(x,z)]$ the interactive computation between A and B, with A's inputs (x,y) and B's inputs (x,z), and uniform randomness; and $[A(x,y;r_A) \leftrightarrow B(x,z;r_B)]$ when we wish to specify randomness. We denote by $\mathsf{VIEW}_P[A(x,y) \leftrightarrow B(x,z)]$ and $\mathsf{OUT}_P[A(x,y) \leftrightarrow B(x,z)]$ the view and output of machine $P \in \{A,B\}$ in this computation. Finally, $\mathsf{TRANS}[A(x,y) \leftrightarrow B(x,z)]$ denotes the *transcript* of the interaction $[A(x,y) \leftrightarrow B(x,z)]$ which consists of all messages exchanged in the computation.

We also assume familiarity with *oracle* Turing machines, which are ordinary TMs with an extra tape called the *oracle communication tape.* An oracle TMs A will be written as $A^{\langle \cdot \rangle}$ to insist that it is an oracle TM; in addition, we write $A^{\mathcal{I}}$ when A's oracle is fixed to $\mathcal{I}$. Recall that each query to $\mathcal{I}$ counts as one step towards the running time of $A^{\mathcal{I}}$.

Unless specified otherwise, all algorithms receive a parameter $n \in \mathbb{N}$, called the security parameter, as their first input. Often, the security parameter will not be mentioned explicitly and dropped from the notation. With some exceptions, all algorithms run in $\mathsf{poly}(n)$ steps and all inputs have $\mathsf{poly}(n)$ length. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if it approaches zero faster than every polynomial.

Two ensembles $\{\mathcal{X}_\backslash\}_{n\in\mathbb{N}}$ and $\{\mathcal{Y}_\backslash\}_{n\in\mathbb{N}}$ are said to be *computationally indistinguishable*, denoted $\{\mathcal{X}_\backslash\} \overset{c}{\approx} \{\mathcal{Y}_\backslash\}$, if for all non-uniform probabilistic polynomial time (PPT) distinguishers D, sufficiently large n, and every advice string z_n: $|\Pr_{x\leftarrow\mathcal{X}_n}[D_n(x) = 1] - \Pr_{y\leftarrow\mathcal{Y}_n}[D_n(y) = 1]| \le \mathsf{negl}(n)$, where we write $D_n(a)$ to denoted $D(n, z_n, a)$, and negl is a negligible function. The statistical distance between two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ over the same support S is denoted by $\Delta(X, Y) = \frac{1}{2}\sum_{a\in S} |\Pr[X = a] - \Pr[Y = a]|$. We say that ensembles $\{\mathcal{X}_n\}_{n\in\mathbb{N}}$ and $\{\mathcal{Y}_n\}_{n\in\mathbb{N}}$ are *statistically indistinguishable* (or statistically close), denoted $\{\mathcal{X}_n\} \overset{s}{\approx} \{\mathcal{Y}_n\}$, if there exists a negligible function negl such that $\Delta(\mathcal{X}_n, \mathcal{Y}_n) \le \mathsf{negl}(n)$ for all sufficiently large n.

Standard primitives. In this work, we will be using a family of *injective* one-way functions. In addition, unless specified otherwise, we assume that all functions $f \in \mathcal{F}_n$ in the family have an *efficiently testable range membership*: i.e., there exists a polynomial time algorithm to test that $y \in \mathsf{Range}(f)$ where $\mathsf{Range}(f)$ denotes the range of f.

We will also be using a family of *collision resistant hash functions* (CRHF) $\{\mathcal{H}_n\}$ where $h : \{0, 1\}^* \to \{0, 1\}^{\mathsf{poly}(n)}$ for $h \in \mathcal{H}_n$; recall that $\{\mathcal{H}_n\}$ is a CRHF family if there exists a negligible function negl such that for every non-uniform PPT machines A, every sufficiently large n, and every advice string z_n: $\Pr_{h\leftarrow\mathcal{H}_n}[h(x) = h(y) : (x, y) \leftarrow A(z_n, h)] \le \mathsf{negl}(n)$.

Finally, we will also be using a non-interactive, perfectly binding *commitment scheme* for committing strings of polynomial length. A commitment to a string m using randomness u will be denoted by $c = \mathsf{Com}(m; u)$. Without loss of generality, we assume that the message m committed to in c can be recovered given the randomness u and the string c. We assume perfectly binding schemes purely for the simplicity of exposition. One can replace Com by the 2-round statistically-binding commitment scheme of Naor [Nao89] without affecting our results.

2.1 Interactive Proofs, Proofs of Knowledge, and Witness Indistinguishability

We recall the standard definitions of interactive proofs [GMR85], witness indistinguishability [FS90], and proofs of knowledge [GMR85, TW87, FFS88, FS90, BG92, PR05b].

Definition 1 (Interactive Proofs). *A pair of probabilistic polynomial time interactive Turing machines $\langle P, V\rangle$ is called an interactive argument system for a language $\mathbf{L} \in \mathbf{NP}$ with witness relation $\mathbf{R}$ if there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ *such that the following two conditions hold:*

– Completeness*: for every $x \in \mathbf{L}$, and every witness w such that $\mathbf{R}(x, w) = 1$, it holds that*

$$\Pr[\mathsf{OUT}_V[P(x, w) \leftrightarrow V(x)] = 1] = 1.$$

– Soundness*: for every $x \notin \mathbf{L}$, every interactive Turing machine P^* running in time at most $\mathsf{poly}(|x|)$, and every $y \in \{0, 1\}^*$,*

$$\Pr[\mathsf{OUT}_V[P^*(x, y) \leftrightarrow V(x)] = 1] \leq \mathsf{negl}(|x|).$$

If the soundness condition holds for every (not necessarily PPT*) machine P^* then $\langle P, V\rangle$ is called an interactive* proof *system.* $\square$

The probability in the soundness condition is called the *soundness error* of the system, and we say that the system has *negligible* soundness error since this probability is at most $\mathsf{negl}(|x|)$. Although, traditionally soundness error is defined in terms of the statement length $|x|$, in cryptographic contexts, it is convenient to define it in terms of the security parameter n, and write $\mathsf{negl}(n)$. This is without loss of generality, since in our setting since $|x| = \mathsf{poly}(n)$. Also, in this work, we will use words "argument" and "proof" interchangeably throughout the paper.

Definition 2 (Proof of Knowledge). *Let $\langle P, V\rangle$ be an interactive proof system for a language $\mathbf{L} \in \mathbf{NP}$ with witness relation $\mathbf{R}$. We say that $\langle P, V\rangle$ is a* proof of knowledge *(*POK*) for relation $\mathbf{R}$ if there exists a polynomial p and a probabilistic oracle machine E (called the* extractor*) such that for every* PPT ITM *P^*, there exists a negligible function* negl *such that for every $x \in \mathbf{L}$, and every $(y, r) \in \{0, 1\}^*$ such that $q_{x,y,r} := \Pr[\mathsf{OUT}_V[P^*_{x,y,r} \leftrightarrow V(x)] = 1] > 0$ where $P^*_{x,y,r}$ denotes the machine P^* whose common input, auxiliary input, and randomness are fixed to x, y and r respectively and the probability is taken over the randomness of V, the following conditions holds:*

– *the expected number of steps taken by $E^{P^*_{x,y,r}}$ is bounded by $\frac{p(|x|)}{q_{x,y,r}}$, where $E^{P^*_{x,y,r}}$ is machine E with oracle access to $P^*_{x,y,r}$;*
– *except with negligible probability, $E^{P^*_{x,y,r}}$ outputs w^* such that $\mathbf{R}(x, w^*) = 1$.* $\square$

Definition 3 (Witness Indistinguishable Proofs). *Let $\langle P, V\rangle$ be an interactive proof system for a language $\mathbf{L} \in \mathbf{NP}$ with witness relation $\mathbf{R}$. We say that $\langle P, V\rangle$ is* witness indistinguishable *(*WI*) for relation $\mathbf{R}$ if for every* PPT ITM *V^*, every statement $x \in \mathbf{L}$, every pair of witnesses (w_1, w_2) such that $\mathbf{R}(x, w_i) = 1$ for every $i \in \{1, 2\}$, and every (advice) string $z \in \{0, 1\}^*$, it holds that $\{\mathsf{VIEW}^{(1)}_{|x|}\} \overset{c}{\approx} \{\mathsf{VIEW}^{(2)}_{|x|}\}$ where $\{\mathsf{VIEW}^{(i)}_{|x|}\} := \mathsf{VIEW}_{V^*}[P(x, w_i) \leftrightarrow V^*(x, z)]$.* $\square$

As before, w.l.o.g., we can replace $|x|$ by the security parameter n in all definitions above. We remark that there exists a WIPOK with *strict* polynomial time extraction in *constant rounds* using non-black-box techniques [BL04] and in $\omega(1)$ rounds using black-box techniques [GMR85, Blu87].

Three round, public-coin WIPOK**and** ZAP**s.** The classical protocols of [GMR85, Blu87], based on the existence of non-interactive perfectly binding commitment schemes, are 3-round *witness indistinguishable, proof of knowledge* (WIPOK) protocols (for every language in **NP**). We will use Blum's protocol [Blu87] as a building block and denote its three messages by $\langle \alpha, \beta, \gamma \rangle$, where β is random string of sufficient length.[7]

A ZAP for a language **L**, introduced by Dwork and Naor [DN00], is a *two round witness indistinguishable* interactive proof for **L**. ZAPs can be constructed from a variety of assumptions such as non-interactive zero-knowledge proofs [BFM88, BSMP91] (which in turn can be based on trapdoor permutations [FLS99]) and verifiable random functions [MRV99]. In fact, even *non-interarctive* (i.e., one round) constructions for ZAPs for all of **NP** exist based on bilinear pairings [GOS06] and derandomization techniques [BOV03].

We will use the two round construction of [DN00] based on NIZK as a building block and denote its two messages by $\langle \sigma, \pi \rangle$ where σ is a randomly string of sufficient length. An important property of this construction is *adaptive soundness*: the statement to be proven can be chosen *after* the string σ has been sent by the verifier. We will rely on this property in our security proofs.

2.2 Concurrent Zero Knowledge

We now recall the notion of concurrent zero-knowledge [DNS98] in which one considers a "concurrent adversary" V^* who interacts in many copies of P, proving adaptively chosen, possibly correlated, polynomially many statements. We follow conventions established in [DNS98, PRS02, Ros04].

Concurrent attack. The *concurrent attack* on an interactive proof systems $\langle P, V \rangle$ for language $\mathbf{L} \in \mathbf{NP}$ with witness relation $\mathbf{R}$ considers an arbitrary interactive TM V^* which opens at most $m = m(n)$ sessions for an arbitrary polynomial m with arbitrary auxiliary input $z \in \{0,1\}^*$. Let $\boldsymbol{x} := \{x_i\} \in \mathbf{L}^m$ be set of statements in $\mathbf{L}$ of length at most $\mathsf{poly}(n)$, and $\boldsymbol{w} := \{w_i\}_{i \in [m]}$ be such that $\mathbf{R}(x_i, w_i) = 1$. The attack proceeds by uniformly fixing the random coins of V^* and initiating its execution on input the security parameter $n \in \mathbb{N}$ and auxiliary input z. At each step, V^* either initiates a new *session*—in which case a new prover instance $P(x_i, w_i)$ with fresh randomness is fixed who interacts with V^* in session i; or V^* schedules the delivery of a message of an existing session in which the corresponding prover instance responds with corresponding message. There is no restriction on how V^* schedules the messages of various sessions. We say that V^* *launches m-concurrent attack* on $\langle P, V \rangle$. The output of the attack consists of the view of V^*, denoted $\mathsf{VIEW}_{V^*}^{\langle P,V \rangle}(n, m, \boldsymbol{x}, \boldsymbol{w}, z)$.

[7] We remark that this protocol has a black-box extractor whose *expected* running time is proportional to the inverse of a cheating prover's success probability. However, there also exist WIPOK with *strict* polynomial time extraction in *constant rounds* using non-black-box techniques [BL04] and in $\omega(1)$ rounds using black-box techniques [GMR85, Blu87].

Definition 4 (Concurrent Zero Knowledge). *We say that an interactive proof system $\langle P, V\rangle$ for a language $\mathbf{L} \in \mathbf{NP}$ (with witness relation $\mathbf{R}$) is* concurrent zero knowledge *if for every polynomial $m : \mathbb{N} \to \mathbb{N}$, every* PPT ITM *V^* launching a m-concurrent attack, there exists a* PPT *machine S_{V^*} such that for every set $\boldsymbol{x} := \{x_i\} \in \mathbf{L}^m$ of statements of length at most* $\mathsf{poly}(n)$, *every $\boldsymbol{w} := \{w_i\}_{i\in[m]}$ such that $\mathbf{R}(x_i, w_i) = 1$, and every auxiliary input $z \in \{0,1\}^*$ it holds that*

$$\left\{S_{V^*}\big(n, \boldsymbol{x}, z\big)\right\}_{n\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{VIEW}_{V^*}^{\langle P,V\rangle}\big(n, m, \boldsymbol{x}, \boldsymbol{w}, z\big)\right\}_{n\in\mathbb{N}}.$$

Machine S_{V^*} is called the simulator. □

In what follows, we will sometimes abuse the notation and write V^* to also mean the *description* of the Turing machine V^*. However, when we want to be explicit about the description of a Turing machine M (including V^*), we will actually write $\mathsf{desc}(M)$. For the simulator, we may sometimes write $S_{V^*}(\cdot) := S(V^*, \cdot)$ to insist that the program of V^* is given as an explicit input to the simulator (and drop n from the notation). Further, we will assume a (unique) session identifier for each session represented by a string of length n; this session identifier can be chosen by V^* so long as it is unique for every session. W.l.o.g. we assume that the all-ones string 1^n (not to be confused with the unary representation of the security parameter) is never used as a session identifier and denotes a special symbol.

3 Differing Input Obfuscation for Turing Machines

In this section, we recall the notion of public-coin differing input obfuscation ($\mathsf{pc\text{-}diO}$) for Turing machines [IPS15]. Let $\mathsf{Steps}(M, x)$ denote the number of steps taken by a TM M on input x; we use the convention that if M does not halt on x then $\mathsf{Steps}(M, x)$ is defined to be the *special symbol* ∞. Let $\mathcal{M} = \{\mathcal{M}_n\}$ denote a parameterized collection of polynomial size and polynomial time TMs, i.e., there exists a global polynomial a such that for every $n \in \mathbb{N}$, every $M \in \mathcal{M}_n$, $|M| \le a(n)$ and $\mathsf{Steps}(M, x) \le a(|x|)$ where x can be of arbitrary polynomial length.

We say that a pair of TMs (M_0, M_1) in the class $\mathcal{M}_n$ (for any n) is compatible if they have the same size and for every x, $\mathsf{Steps}(M_0, x) = \mathsf{Steps}(M_1, x)$.

Definition 5 (Compatible TMs). *A pair of Turing machines $(M_0, M_1) \in \mathcal{M}_n \times \mathcal{M}_n$ for $n \in \mathbb{N}$ is said to be* compatible *if $|M_0| = |M_1|$ and for every string $x \in \{0,1\}^*$ it holds that $\mathsf{Steps}(M_0, x) = \mathsf{Steps}(M_1, x)$.*

We remark that the notion of compatible TMs allows the obfuscation to leak the running time of the obfuscated TMs. This is standard requirement; we can also use the convention of [ABG+13, IPS15] where the TMs also output their running time in addition to the “official” output.

We now recall the notion of public-coin differing inputs sampler [IPS15]. Roughly speaking, Samp is public-coin differing-inputs sampler if, on input the random coins z, it output a pair of compatible TMs (M_0, M_1) such that no PPT adversary A having access to z can produce an x such that: $M_0(x) \neq M_1(x)$. We use a slightly different notation form [IPS15], and require that in addition to outputting M_0, M_1, Samp also outputs its random coins. The randomness z of Samp will then not be mentioned as an explicit input. The definition follows.

Definition 6 (Public-coin Differing-Inputs Sampler for TMs). *We say that a (possibly non-uniform)* PPT *Turing machine* Samp *is a* public-coin differing-inputs sampler for Turing machines *if the following conditions hold:*

1. *the output of* $\mathsf{Samp}(1^n)$ *is a triplet* (z, M_0, M_1) *such that* z *is the randomness of* Samp *and* $(M_0, M_1) \in \mathcal{M}_n \times \mathcal{M}_n$ *is* always *a pair of* compatible TM*s;*
2. *for every (possibly non-uniform)* PPT TM A *there exists a negligible function* negl *such that for all* $n \in \mathbb{N}$*:*

$$\Pr\left[M_0(x) \neq M_1(x) : (z, M_0, M_1) \leftarrow \mathsf{Samp}(1^n; z)\ ;\ A(z, M_0, M_1) = x\right] \leq \mathsf{negl}(n).$$

For convenience, a public-coin differing-input sampler will also be referred to as a **nice sampler**. □

Public-coin differing-input obfuscator. We now present the definition of a *public-coin differing input obfuscator* for Turing machines. Roughly speaking, the notion states that a machine $\mathcal{O}$ is a $\mathsf{pc\text{-}diO}$ if the following holds: if there exists a PPT distinguisher D who distinguishes $\mathcal{O}(M_0)$ from $\mathcal{O}(M_1)$ when given as auxiliary input the random coins z of the sampler who samples (M_0, M_1), then it is easy to find an x (given z) such that $M_0(x) \neq M_1(x)$. In other words, if it is hard to find the "differing input" x then the two obfuscations are indistinguishable.

Definition 7 (Public-coin Differing-Inputs Obfuscator for Turing Machines, [IPS15]). *A uniform* PPT *machine* $\mathcal{O}$ *is called a* public-coin differing input obfuscator ($\mathsf{pc\text{-}diO}$) *for a class of Turing machines* $\{\mathcal{M}_n\}$ *if the following conditions are satisfied:*

1. Polynomial slowdown and functionality: *there exists a polynomial* a_{dio} *such that for every* $n \in \mathbb{N}$*, every* $M \in \mathcal{M}_n$*, every input* x*, and every* $\widetilde{M} \leftarrow \mathcal{O}(n, M)$*, the following conditions hold:*
   – $\mathsf{Steps}(\widetilde{M}, x) \leq a_{\text{dio}}\Big(n,\ \mathsf{Steps}(M, x)\Big)$
   – $\widetilde{M}(x) = M(x)$

   Polynomial a_{dio} *is called the* slowdown polynomial *of* $\mathcal{O}$.
2. Indistinguishability: *for every public-coin differing-input (a.k.a. nice) sampler* Samp *(i.e., satisfying definition 6), for every (possibly non-uniform)*

PPT *distinguisher D, there exists a negligible function* negl *such that for all n:*

$$\Big| \Pr\left[D\left(z, \mathcal{O}(n, M_0)\right) = 1 : (z, M_0, M_1) \leftarrow \mathsf{Samp}\left(1^n; z\right)\right]$$
$$- \Pr\left[D\left(z, \mathcal{O}(n, M_1)\right) = 1 : (z, M_0, M_1) \leftarrow \mathsf{Samp}\left(1^n; z\right)\right] \Big| \leq \ \mathsf{negl}(n).$$

where the probability is taken over the randomness of both Samp *and* $\mathcal{O}$. □

Candidate constructions. In [IPS15], a candidate construction of pc-diO for all polynomial time TMs with variable length input of polynomial size is presented. The functionality of the construction allows the TMs to accept inputs of any length, even larger than polynomial. The security states that if a PPT machine distinguishes the obfuscation of the given TMs, there will exist an input of *polynomial* size, which can be extracted, such that the two machines will differ on that input. The assumptions underlying this construction are: pc-diO for NC^1 circuits, fully homomorphic encryption, and SNARKs for **NP**. The construction of [GGH+13] is seen as a plausible candidate for pc-diO for NC^1 and existing implausibility results [GGHW14] are unlikely to have a consequence to this assumption. Construction of diO with stronger forms of auxiliary input—worst case in which the security must hold for all auxiliary strings μ, and distributional where it holds for specific distributions of μ—were presented in [ABG+13, BCP14].

4 Constant Round Concurrent Zero-knowledge

The simplest way to see why the protocol of previous section does not work in the concurrent setting is to consider its execution in a recursively interleaved schedule (described by Dwork, Naor, and Sahai [DNS98]). In the context of our protocol, this schedule will have n sessions interleaved recursively as follows: session n does not "contain" any messages of any other session, and all messages of session i are contained between messages c_{i-1} and r_{i-1} of session $i-1$ for every i , starting from $i = n$. A pictorial representation of this scheduling is given in the full version [PPS15]. The double-headed arrows marked by π_i represent the rest of the messages of the i-th session. Roughly speaking, the simulation fails because of the following: in order to simulate session i, the simulator needs to extract the secret s_i by running the program $\widetilde{M}_{\lambda_i,s_i}$; however, the execution of $\widetilde{M}_{\lambda_i,s_i}$ contains an execution of $\widetilde{M}_{\lambda_{i+1},s_{i+1}}$ and due to this recursion, simulator's total running time in session 1 is exponential in n.

Formally, let $t_3 \geq 1$ be the time taken by the verifier in computing r_3 on input the string c_3. Then clearly, the time taken by the simulator in running the obfuscated machine $\widetilde{M}_{\lambda_3,s_3}$ is $T_3 \geq t_3$. Then, if t_2 denotes the time taken by the simulator to obtain string r_2, we have that $t_2 \geq t_3 + T_3 \geq 2t_3$. Clearly, the time taken by the simulator to extract s_2 by running the program $\widetilde{M}_{\lambda_2,s_2}$ will be at least $T_2 \geq t_2 \geq 2t_3$. By repeating this argument for session 1, we have that

$T_1 \geq t_1 \geq t_2 + T_2 \geq 2t_2 \geq 2^2 t_3$. Repeating this argument for n sessions in the DNS schedule, the total time taken by the simulator will be $\geq 2^{n-1}$.

Avoiding recursive computation via DGS-oracle. It is clear that the reason our stand-alone simulator runs in exponential time is because in order to compute s_i for session i, the simulator runs (the obfuscation of) a program which recursively runs such a program for every interleaved session between c_i and r_i. That is, the program $\widetilde{M}_{\lambda_i, s_i}$ ends up *recomputing* all of the secrets of the interleaved sessions even though they have already been computed.

We can avoid this recomputation as follows. Let $\mathcal{I}$ be an oracle which takes as input queries of the form $(f, \widetilde{s})$—where f is an *injective* one-way function and $\widetilde{s}$ is in the range of f—and returns the unique value s such that $f(s) = \widetilde{s}$.[8] Now consider an arbitrary program $\Pi^{\mathcal{I}}$ which has access to the inversion oracle $\mathcal{I}$. Clearly, if r is chosen randomly, then for any (fixed) program $\Pi^{\mathcal{I}}$ and any fixed input a, the probability that $\Pi^{\mathcal{I}}(a) = r$ is at most 2^{-n}. This is because once the description of the oracle program $\Pi^{\langle\cdot\rangle}$ is fixed, the output of $\Pi^{\mathcal{I}}(a)$ is deterministically fixed (for any fixed input a chosen prior to seeing r) and r hits this value with probability at most 2^{-n}.

Our main point here is that it is hard to come up with a satisfying "fake witness" ω to the transcripts $\lambda = \langle h, c, r\rangle$ *even if* the program committed in c is given access to the inversion oracle $\mathcal{I}$. On the other hand, the simulator can still predict r as before. However, more importantly, by means of the oracle $\mathcal{I}$ we can avoid the recursive re-computation of the secrets in the concurrent setting as follows.

Consider an *alternative* simulator $S^{\langle\cdot\rangle}$ which will be given access to the oracle $\mathcal{I}$. This simulator will have access to both, the program of the verifier V^* as well as *its own program*, given as explicit inputs, collectively denoted as $\Pi^{\langle\cdot\rangle}_{S,V^*}$. The simulator, on input a session index i, will work by initiating an execution of V^*. It will commit to program $\Pi^{\langle\cdot\rangle}_{S,V^*}(j)$ in session j (ignoring for the moment the fact that simulator needs fresh randomness); finally, this simulator *does not run any obfuscated program* to compute the secrets. Instead it queries the oracle $\mathcal{I}$ on "well formed" $(f_j, \widetilde{s}_j)$ for every session $j \neq i$; when $j = i$ it simply returns the string r_i. Then, if all goes well, observe that program $\Pi^{\langle\cdot\rangle}(i)$ predicts string r_i in polynomial time (given $\mathcal{I}$) and this holds for every session i. In particular, there is no recursive recomputation of the secrets since they can be fed to the program directly once they have been computed. We note that such an oracle was first used by Deng, Goyal, and Sahai [DGS09] to construct the first *resettably-sound resettable zero-knowlege* protocol for **NP**.

It should be clear that the actual simulation will be performed by a "main" simulator S_{main} which will *not have access to any* inversion oracle, and run in (strict) polynomial time. The main simulator will run in the same manner as the alternative simulator $S^{\langle\cdot\rangle}$ except that instead of using $\mathcal{I}$, it will run the

[8] We assume that it is easy to test that f is injective and that $\widetilde{s}$ is in the range of f. These requirements are only for simplicity and the protocol works even if it is not easy to test these properties.

obfuscated programs (only once for each session) to recover the secrets. To ensure efficient simulation, once a session secret has been recovered, it will be stored in a global table $\mathcal{T}$ (which will be used to simulate answers of $\mathcal{I}$). Therefore the "fake witness" will now have the form $\omega = \langle u, \mathit{\Pi}^{\langle\cdot\rangle}, \mathcal{T}\rangle$, but the statements will still have the same form $\lambda = \langle h, c, r\rangle$; and we require that $\mathit{\Pi}^{\mathcal{T}}$ outputs r within finite steps (see details below).

Relation $\mathbf{R}_{\text{sim}}$ and the simple variant of our protocol. To formally capture the above mentioned requirement for the transcripts λ, we define a relation $\mathbf{R}_{\text{sim}}$ in figure 1. The family of injective one-way functions is denoted by $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ and that of collision-resistant hash functions by $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$. An important observation regrading $\mathbf{R}_{\text{sim}}$ is that since table $\mathcal{T}$ is not a part of the commitment c (and it should not be), we must enforce that $\mathit{\Pi}^{\langle\cdot\rangle}$ *never makes any invalid queries to* $\mathcal{T}$. This is because after seeing r, it is easy to design a "bad" table $\mathcal{T}$ which will encode r by means of "bad" entries and "satisy" λ.

Relation $\mathbf{R}_{\text{sim}}$ allows us to prove that no efficient prover can compute ω such that $\mathbf{R}_{\text{sim}}(\lambda, \omega) = 1$ with noticeable probability where λ is the transcript of $\mathsf{GenStat}$ with an honest verifier. We prove this claim formally in lemma 1 under the collision-intractability of $\{H_n\}$. We note that $\mathbf{R}_{\text{sim}}$ is not decidable in polynomial time in general, but this will not be an issue for our reductions since we will ensure that it is checked only on "good" instances (which can be verified in polynomial time).

Instance: A tuple $\langle h, c, r\rangle \in \mathcal{H}_n \times \{0,1\}^{\mathsf{poly}(n)} \times \{0,1\}^n$ where $h : \{0,1\}^* \rightarrow \{0,1\}^n$.

Witness: A tuple $\langle u, \mathit{\Pi}^{\langle\cdot\rangle}, \mathcal{T}\rangle \in \{0,1\}^{\mathsf{poly}(n)} \times \{0,1\}^* \times \{0,1\}^*$ where $\mathit{\Pi}^{\langle\cdot\rangle}$ is an *oracle* Turing machine, and $\mathcal{T}$ is a table containing entries of the form $(f, \widetilde{s}, s)$ such that when queried on $(f, \widetilde{s})$, $\mathcal{T}$ returns s, denoted $\mathcal{T}(f, \widetilde{s}) = s$.

Relation: $\mathbf{R}_{\text{sim}}\big(\langle h, c, r\rangle,\ \langle u, \mathit{\Pi}^{\langle\cdot\rangle}, \mathcal{T}\rangle\big) = 1$ if and only if all of the following conditions hold:

1. $c = \mathsf{Com}\left(h\left(\mathit{\Pi}^{\langle\cdot\rangle}\right)\ ;\ u\right)$
2. $\forall\ (f, \widetilde{s}, s) \in \mathcal{T}$ it holds that $f \in \mathcal{F}_n$ is an *injective* function and $f(s) = \widetilde{s}$
3. Program $\mathit{\Pi}^{\mathcal{T}}$, takes no input, outputs the string r, and halts within 2^n steps.
4. Program $\mathit{\Pi}^{\mathcal{T}}$ makes oracle queries of the form $(f, \widetilde{s})$ such that:

$$\forall \text{ queries } (f, \widetilde{s})\ \ \exists\, s \ \text{ s.t. }\ (f, \widetilde{s}, s) \in \mathcal{T}$$

Fig. 1. Relation $\mathbf{R}_{\text{sim}}$ based on a perfectly binding commitment Com.

We are now ready to describe the simpler variant of our protocol which is constant round and concurrent zero-knowledge. The protocol has three stages: in first stage λ is sampled, in second stage the verifier sends $(f, \widetilde{s})$ and *Turing machine* $\widetilde{M}_{\lambda,s}$ (and also proves that is is a correctly generated), in stage 3, P proves the knowledge of either a witness to x (the statement) or s s.t. $f(s) = \widetilde{s}$. The formal description of our protocol, named Simple-c$\mathcal{ZK}$, appears in figure 2. In the protocol description, we have renamed the machine $M_{\lambda,s}$ (which was only informally stated earlier) to $\mathsf{SimLock}_{\lambda,s}(\cdot) := \mathsf{SimLock}(\lambda, \cdot, s)$[9] where, formally:

$\underline{\mathsf{SimLock}(\lambda, \omega, s)}$**:**

- Test if $\mathbf{R}_{\text{sim}}(\lambda, \omega) = 1$, and if so output s; else, output 0^n.

Inputs. The common input to P and V is a statement $x \in \mathbf{L}$ where language $\mathbf{L} \in \mathbf{NP}$. The prover's auxiliary input is a witness w such that $\mathbf{R}(x, w) = 1$. The security parameter n is an implicit input to both parties.

Protocol. The protocol proceeds in three stages.

Stage 1: P and V execute the GenStat protocol in which V sends the first message $h \leftarrow \mathcal{H}_n$, P sends the second message $c = \mathsf{Com}(0^n; u)$ for a random u, and V sends the final message $r \leftarrow \{0,1\}^n$. Let $\lambda = \langle h, c, r \rangle$ be the transcript.

Stage 2: V samples an injective one-way functions $f \leftarrow \mathcal{F}_n$, a random input $s \in \{0,1\}^n$, and a sufficiently long random tape $\zeta \in \{0,1\}^{\mathsf{poly}(n)}$ and computes:

$$\widetilde{s} = f(s), \qquad \widetilde{M}_{\lambda,s} \leftarrow \mathcal{O}\left(\, \mathsf{SimLock}_{\lambda,s} \ ;\ \zeta \right) \tag{1}$$

V sends $(f, \widetilde{s}, \widetilde{M}_{\lambda,s})$, and proves using a constant round ZK protocol (say Π_{ZK}) that there exist (s, ζ) satisfying equation (1) above.

Stage 3: P proves to V, using a 3-round WIPOK (say Π_{WIPOK}) the knowledge of either:

- w such that $\mathbf{R}(x, w) = 1$; OR
- s such that $f(s) = \widetilde{s}$.

***Verifier's output*:** V accepts if the proof in stage 3 succeeds; otherwise, it rejects.

Fig. 2. The simpler variant of our protocol: Simple-c$\mathcal{ZK}$.

Relation $\mathbf{R}^a_{\text{sim}}$, language $\mathbf{L}^a_{\text{sim}}$. Relation $\mathbf{R}_{\text{sim}}$ is undecidable in polynomial in general. We define a polynomial time decidable version of $\mathbf{R}_{\text{sim}}$. For a polynomial $a : \mathbb{N} \to \mathbb{N}$, relation $\mathbf{R}^a_{\text{sim}}$ is defined as follows: $\mathbf{R}^a_{\text{sim}}$ is identical to $\mathbf{R}_{\text{sim}}$

[9] SimLock stands for "simulator's lock," i.e., only the simulator will be able to "unlock" the secret s from this program.

except that the witness $(u, \Pi^{\langle\cdot\rangle}, \mathcal{T})$ satisfies following additional constraints: (1) $|\mathcal{T}| \leq a(n)$, and (2) $\Pi^{\mathcal{T}}$ halts in at most $a(n)$ steps.

We define $\mathbf{L}_{\text{sim}}$ and $\mathbf{L}^a_{\text{sim}}$ to be the languages corresponding to $\mathbf{R}_{\text{sim}}$ and $\mathbf{R}^a_{\text{sim}}$ respectively. Note that for every polynomial a, it holds that $\mathbf{L}^a_{\text{sim}} \in \mathbf{NP}$. We say that $\mathcal{Z} = \{\mathcal{Z}_n\}$ is a *hard distribution* over the statements of $\mathbf{L}^a_{\text{sim}}$ if there exists a negligible function negl such that for every non-uniform PPT algorithm A^* and every sufficiently large n it holds that $\Pr[\lambda \leftarrow \mathcal{Z}_n; \omega \leftarrow A^*(1^n, \lambda); \mathbf{R}^a_{\text{sim}}(\lambda, \omega) = 1] \leq \mathsf{negl}(n)$.

The main result of this section is the following theorem.

Theorem 1. *Assume the existence of collision-resistant hash functions and injective one-way functions. Further,* public-coin differing-inputs obfuscation (pc-diO) *for the class of all polynomial-size Turing machines that halt in a polynomial number of steps.*[10] *Then, there exists a constant round, fully concurrent zero-knowledge protocol with negligible soundness, for all languages in* $\mathbf{NP}$.

We prove the above theorem by proving that protocol Simple-c$\mathcal{ZK}$ is a fully concurrent zero-knowledge protocol with negligible soundness error. It is clear that the protocol has constant rounds and perfect completeness. We have already discussed briefly the main ideas for proving the soundness and concurrent ZK of this protocol. We discuss a few more points here and provide the full proofs in [PPS15].

To prove the soundness, we start by proving some claims about the obfuscation of Turing machine $\mathsf{SimLock}_{\lambda,s}$. In Section 6 we prove that it is hard for any (non-uniform) prover P^* to write a "fake witness" ω to the statements λ sampled using the preamble GenStat (see lemma 1). Using this lemma, we show that a sampling algorithm that outputs the pair of machines $(\mathsf{SimLock}_{\lambda,s}, \mathsf{SimLock}_{\lambda,0^n})$ is a *nice* sampler for Turing machines—which, roughly speaking, means that it is hard to produce an input y such that $\mathsf{SimLock}_{\lambda,s}(y) \neq \mathsf{SimLock}_{\lambda,0^n}(y)$. Therefore, by security of pc-diO, the obfuscation of $\mathsf{SimLock}_{\lambda,s}$ will be indistinguishable from that of $\mathsf{SimLock}_{\lambda,0^n}$. This however requires some care since we have to ensure that λ can indeed be correctly sampled using public-coins. But λ consists of (h, r) which are completely random strings, and c is the output of P^* which is a publicly known deterministic TM. Therefore we can actually sample λ and still ensure hardness of finding a differing-input given (h, r). Now, the soundness follows by considering three hybrids as before and violating the hardness of one-way functions (similar to the soundness of standalone ZK in section 1.1). The full proof appears in [PPS15].

To prove concurrent ZK we consider two simulators. The first one is called the internal simulator which requires access to an inversion oracle $\mathcal{I}$ for (injective) one-way functions. The second is the main simulator which essentially runs exactly as the internal simulator (by committing its description in c) and extracting the secrets using the obfuscated programs. The full descriptions of both

[10] We note that we actually do not need obfuscation for the class of all PPT Turing machines. Instead, we only need obfuscation for those Turing machines of the form $\mathsf{SimLock}^a$ where a is a polynomial and $\mathsf{SimLock}^a$ is the same as $\mathsf{SimLock}$ except that it runs for at most $a(|x|)$ steps on input x.

the simulators as well as the full proof of indistinguishability of simulation are given in [PPS15].

An important issue that we did not discuss relates to the randomness used in the simulation. For the simulation to work, it is essential the internal simulator and the main simulator must use identical randomness in computing the messages that are "fed" to the verifier. This creates a circularity in the security proof: how can the commitments sent by the main simulator be "secure" when the message in the commitment (i.e., the program of the internal simulator) is correlated to the randomness used to create the commitment. We address this issue as follows: we do not include the randomness as part of the internal simulator's description in the "plain;" instead, we include it in the "committed" form using a perfectly binding commitment *which can be recovered using the inversion oracle $\mathcal{I}$*—e.g., using commitments based on hard-core bits [GL89].

5 The Four Round Construction

In the previous section, we presented a reduction from *constant round, concurrent zero-knowledge* to diO based on standard cryptographic assumptions. In this section, we present a similar reduction for four message concurrent zero-knowledge.

Let us start by optimizing the number of rounds in our constant round protocol of previous section. The standalone ZK protocol used in stage 2 has at least four rounds.[11] Since the last message of this ZK protocol must come from the verifier, our resulting protocol will have at least five rounds even after optimizations.

We consider two approaches to obtain a four round protocol. First, we can use a two-round ZK protocol with *super polynomial time simulation*[Pas03]. This approach gives us a reduction where the soundness of the resulting protocol must assume sub-exponential hardness assumptions. The second approach is to use a WI protocol to prove the correctness of the obfuscated program. However, in typical applications of WI, to get any useful security we must somehow ensure that the statement being proven has at least two witnesses.

The standard approach in such cases is to consider two independently sampled statements, in this case, two obfuscated programs $\widetilde{M}_{\lambda,s}$ and $\widetilde{M}_{\lambda,s'}$; and prove that at least one of them is correctly constructed using a WI proof. However, this approach actually fails for a very interesting reason. Although it does hide one of the secrets s, s', it actually breaks the simulation. Indeed, the internal simulator committed to in the preamble, will have no efficient way of knowing which of these two programs is actually correctly prepared. In particular, it will have to ask for the inversion of *two* challenges per session but the main simulator might be able to return only one of them (since one of the obfuscated programs could have been maliciously prepared). Attempting to overcome this subtle issue actually breaks the hardness of $\mathbf{R}_{\mathrm{sim}}$.

[11] To keep our *reduction* from concurrent ZK to obfuscation free from "knowledge assumptions," we cannot use 3-round ZK protocols based on such assumptions.

We therefore use a different approach; we set up an "intermediate statement" which is selected by the prover, and require the prover to provide a WIPOK of its correctness. The verifier then proves that either this intermediate statement is true or the obfuscated program is correctly prepared. The intermediate statement is prepared in such a way that it is possible to make it false and succeed (using the real witness for x) without the verifier noticing. This allows us to ensure that the obfuscated program must be correctly prepared and simulation still continues to go through. For the soundness, roughly speaking, we can extract the witness corresponding to the "intermediate statement" by using the extractor of WIPOK; we then use it to simulate the WI proof that comes from verifier's side. This allows us to again enforce the ideas we developed to prove the soundness of the $\mathsf{Simple\text{-}c}\mathcal{ZK}$ protocol.

To setup the "intermediate statement" we use perfectly binding commitments to specially prepared strings. In the final proof, we will need to actually extract the secret s to violate the hardness of one-way functions. We get around this difficulty by using a combination of the WIPOK used by the prover and a ZAP proof. We now present a sketch of our four round protocol below. The formal description of the protocol appears [PPS15].

Four round protocol for concurrent zero-knowledge. The protocol has four components whose messages will be sent in parallel:

1. The first component is the $\mathsf{GenStat}$ protocol, producing statements of the form $\lambda = \langle h, c, r\rangle$.
2. The second component is a three round WIPOK given by the *prover to the verfier*. The prover prepares two commitments, namely $\widetilde{t}_1 = \mathsf{Com}(0 \parallel t_1; v_1)$ and $\widetilde{t}_2 = \mathsf{Com}(0 \parallel t_2; v_2)$ and proves that either $(\widetilde{t}_1, \widetilde{t}_2)$ are correctly prepared or x is true. The 3 messages of this WIPOK will be denoted by $\langle \alpha, \beta, \gamma\rangle$.
3. The final component is a ZAP for a specially prepared statement, which will let us extract either a witness to x or the secret s in the proof of soundness. The special statement is prepared as follows.
   The prover creates two commitments τ_1, τ_2 such that τ_1 uses string t_1 (defined above in item 2) as its randomness; likewise τ_2 uses t_2. Further, the value committed to in one of them is the witness w for statement x. The prover then proves, using a ZAP, that there exists $i \in \{1, 2\}$ such that τ_i is a commitment to w using t_i. The two messages of this ZAP are denoted by $\langle \sigma', \pi'\rangle$.

To get four rounds, the messages of these components are piggy backed with each other. The main result of this section is the following theorem.

Theorem 2. *Assume the existence of collision-resistant hash functions and trapdoor one-way permutations (alternatively, injective one-way functions and* ZAP *proofs for* $\mathbf{NP}$*). Further, for every polynomial* $a : \mathbb{N} \rightarrow \mathbb{N}$*, and every* hard distribution $\mathcal{Z}$ *over the statements of* $\mathbf{L}^a_{\mathrm{sim}}$*, assume the existence of* $\mathcal{Z}$-auxiliary differing-input obfuscation *(*diO*) for the class of all polynomial-size Turing machines that halt in a polynomial number of steps. Then, there exists a* four message*, fully concurrent zero-knowledge protocol with negligible soundness, for all languages in* **NP***.*

We have already discussed main ideas behind the proof of soundness and concurrent zero-knowledge of this protocol. The full details are given in [PPS15].

References

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013, 2013.

[App13] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. Cryptology ePrint Archive, Report 2013/699, 2013. `http://eprint.iacr.org/2013/699.pdf`.

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

[Bar02] Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *FOCS*, 2002.

[BBC$^+$14] Boaz Barak, Nir Bitansky, Ran Canetti, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Obfuscation for evasive functions. In *TCC*, 2014. Preliminary version on Eprint 2013: `http://eprint.iacr.org/2013/668.pdf`.

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

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCG$^+$11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, pages 722–739, 2011.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. Extractable obfuscation and applications. In *TCC*, 2014. Preliminary version on Eprint 2013: `http://eprint.iacr.org/2013/650.pdf`.

[BCPR13] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. More on the impossibility of virtual-black-box obfuscation with auxiliary input. Cryptology ePrint Archive, Report 2013/701, 2013. `http://eprint.iacr.org/2013/701.pdf`.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Annual IEEE Conference on Computational Complexity (CCC)*, volume 17, 2002. Preliminary full version available as Cryptology ePrint Archive, Report 2001/105.

[BGGL01] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS 2001*, pages 116–125, 2001.

[BGI$^+$01] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. In *Crypto '01*, pages 1–18, 2001.

[BGK$^+$13] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. *IACR Cryptology ePrint Archive*, 2013:631, 2013.

[BGT14] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. `http://eprint.iacr.org/`.

[BL04] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. *SIAM Journal on Computing*, 33(4):783–818, August 2004. Extended abstract appeared in STOC 2002.

[Blu87] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1987.

[BOV03] Boaz Barak, Shien Jin Ong, and Salil P. Vadhan. Derandomization in cryptography. In *CRYPTO*, pages 299–315, 2003.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, pages 273–289, 2004.

[BP12a] Nir Bitansky and Omer Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, pages 223–232, 2012.

[BP12b] Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In *TCC*, pages 190–208, 2012.

[BP13a] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, pages 241–250, 2013.

[BP13b] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/2013/703.pdf`.

[BR13a] Zvika Brakerski and Guy N. Rothblum. Obfuscating conjunctions. In *CRYPTO (2)*, pages 416–434, 2013.

[BR13b] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. Cryptology ePrint Archive, Report 2013/563, 2013. `http://eprint.iacr.org/2013/563.pdf`.

[BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.

[BZ13] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/642, 2013. `http://eprint.iacr.org/`.

[CD09] Ran Canetti and Ronny Ramzi Dakdouk. Towards a theory of extractable functions. In *TCC*, pages 595–613, 2009.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proc. 32th STOC*, pages 235–244, 2000.

[CHJV14] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769, 2014. `http://eprint.iacr.org/`.

[CKPR03] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM Journal on Computing*, 32(1):1–47, February 2003. Preliminary version in STOC '01.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010. Full version: `http://www.cs.cornell.edu/~rafael/papers/ccacommit.pdf`.

[CLP13a] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

[CLP13b] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from p-certificates. In *FOCS*, 2013.

[COPV13] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettability from one-way functions. In *FOCS*, pages 231–240, 2013.

[CPS13] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. In *STOC*, pages 231–240, 2013.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, pages 72–89, 2010.

[CV13] Ran Canetti and Vinod Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. *IACR Cryptology ePrint Archive*, 2013:500, 2013.

[Dam91] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, pages 445–456, 1991.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT*, pages 418–430, 2000.

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

[DL07] Yi Deng and Dongdai Lin. Instance-dependent verifiable random functions and their application to simultaneous resettability. In *EUROCRYPT*, pages 148–168, 2007.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Proc. 41st FOCS*, pages 283–293, 2000.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero knowledge. In *Proc. 30th STOC*, pages 409–418, 1998.

[DS98] Cynthia Dwork and Amit Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In *CRYPTO*, pages 442–457, 1998.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. Preliminary version in STOC 1987.

[FLS99] Feige, Lapidot, and Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29, 1999.

[FS89] U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–545, 1989.

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proc. 22nd STOC*, pages 416–426, 1990.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *Advances in Cryptology - CRYPTO 2014*

- *34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 518–535, 2014.

[GIS$^+$10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.

[GJO$^+$13] Vipul Goyal, Abhishek Jain, Rafail Ostrovsky, Silas Richelson, and Ivan Visconti. Concurrent zero knowledge in the bounded player model. In *TCC*, pages 60–79, 2013.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage resilient zero knowledge. In *Advances in Cryptology – CRYPTO*, 2011. Full version at: `http://www.cs.ucla.edu/~abhishek/papers/lrzk.pdf`.

[GK96] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996. Preliminary version appeared in ICALP' 90.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.

[GK13] Shafi Goldwasser and Yael Tauman Kalai. A note on the impossibility of obfuscation with auxiliary input. Cryptology ePrint Archive, Report 2013/665, 2013. `http://eprint.iacr.org/2013/665.pdf`.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st STOC*, pages 25–32, 1989.

[GLP$^+$15] Vipul Goyal, Huijia Lin, Omkant Pandey, Rafael Pass, and Amit Sahai. Round-efficient concurrently composable secure computation via a robust extraction lemma. In *TCC*, 2015. Full version of this work available as IACR Eprint Report 2012/652.

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

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. 17th STOC*, pages 291–304, Providence, 1985. ACM.

[Gol02] Oded Goldreich. Concurrent zero-knowledge with timing, revisited. In *Proc. 34th STOC*, pages 332–340, 2002.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In *CRYPTO*, pages 97–111, 2006.

[Goy13] Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In *STOC*, pages 221–230, 2013.

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

[GS12a] Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In *CRYPTO*, pages 105–123, 2012.

[GS12b] Divya Gupta and Amit Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. *CoRR*, abs/1210.3719, 2012.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *AsiaCrypt '00*, pages 443–457, 2000.

[Had10] Satoshi Hada. Secure obfuscation for encrypted signatures. In *EUROCRYPT*, pages 92–112, 2010.

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

[HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2013/509, 2013. `http://eprint.iacr.org/2013/509.pdf`.

[HT99] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. Cryptology ePrint Archive, Report 1999/009, 1999. `http://eprint.iacr.org/`.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public Coin Differing-Inputs Obfuscation. In *TCC*, 2015. Cryptology Eprint Archive Report 2014/942.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc. 24th STOC*, pages 723–732, 1992.

[Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In *Crypto '95*, pages 311–324, 1995.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-logarithm rounds. In *STOC*, pages 560–569, 2001.

[KPR98] J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the Internet. In *Proc. 39th FOCS*, pages 484–492, 1998.

[KRW13] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. Cryptology ePrint Archive, Report 2013/683, 2013. `http://eprint.iacr.org/2013/683.pdf`.

[LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. `http://eprint.iacr.org/`.

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In *EUROCRYPT*, pages 20–39, 2004.

[Mic94] S. Micali. CS proofs. In *Proc. 35th FOCS*, pages 436–453, 1994.

[MO13] Antonio Marcedone and Claudio Orlandi. Obfuscation ==¿ (ind-cpa security =/=¿ circular security). Cryptology ePrint Archive, Report 2013/690, 2013. `http://eprint.iacr.org/2013/690.pdf`.

[MR13] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. Cryptology ePrint Archive, Report 2013/643, 2013. `http://eprint.iacr.org/2013/643.pdf`.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.

[Nao89] Moni Naor. Bit commitment using pseudo-randomness (extended abstract). In *CRYPTO*, pages 128–136, 1989.

[Pan14] Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In *TCC*, 2014. Preliminary version on Eprint 2012: `http://eprint.iacr.org/2012/362.pdf`.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Eurocrypt '03*, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proc. 36th STOC*, pages 232–241, 2004.

[PPS15] Omkant Pandey, Manoj Prabhakaran, and Amit Sahai. Obfuscation-based non-black-box simulation and four message concurrent zero knowledge for np. In *TCC*, 2015. Full version of this work available as Cryptology ePrint Archive Report 2013/754.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *Proc. 44th FOCS*, 2003.

[PR05a] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, 2005.

[PR05b] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC*, 2005.

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

[PTV10] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Eye for an eye: Efficient concurrent zero-knowledge in the timing model. In *TCC*, pages 518–534, 2010.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In *TCC*, pages 553–570, 2008.

[RK99] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *Eurocrypt '99*, pages 415–432, 1999.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *Crypto '00*, pages 451–468, 2000.

[Ros04] Alon Rosen. *The Round-Complexity of Black-Box Concurrent Zero-Knowledge*. PhD thesis, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 2004.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.

[TW87] M. Tompa and H. Woll. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In *Proc. 28th FOCS*, pages 472–482, 1987.

[Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005.

6 Hardness of GenStat and A Nice Sampler

In this section, we prove that a randomly sampled transcript of GenStat is a hard distribution over the statements of $\mathbf{L}^a_{\text{sim}}$ for every polynomial a. Recall that $\mathcal{Z} = \{\mathcal{Z}_n\}$ is a *hard distribution* over the statements of $\mathbf{L}^a_{\text{sim}}$ if there exists a negligible function negl such that for every non-uniform PPT algorithm A^* and every sufficiently large n it holds that $\Pr[\lambda \leftarrow \mathcal{Z}_n; \omega \leftarrow A^*(1^n, \lambda); \mathbf{R}^a_{\text{sim}}(\lambda, \omega) = 1] \leq \mathsf{negl}(n)$. The preamble GenStat, is recalled below. For convenience, we use a non-interactive perfectly binding commitment scheme; the two-round statistically-binding commitment scheme of [Nao89] also works.

6.1 Preamble GenStat

Statement generation protocol. Let $\{\mathcal{H}_n\}$ be a family of collision-resistant hash functions (CRHF) $h \in \mathcal{H}_n$ such that $h : \{0,1\}^* \to \{0,1\}^n$ and Com be a non-interactive perfectly-binding commitment scheme for $\{0,1\}^n$. The statement generation protocol GenStat $:= \langle P_1, V_1 \rangle$ is a three round protocol between P_1 and V_1 which proceeds as follows:

Protocol GenStat $:= \langle P_1, V_1 \rangle$**:**
1. V_1 sends a random $h \leftarrow \mathcal{H}_n$
2. P_1 sends a commitment $c = \mathsf{Com}(0^n; u)$ where u is a randomly chosen
3. V_1 sends a random string $r \leftarrow \{0,1\}^n$

The transcript of the protocol is $\lambda := \langle h, c, r \rangle$. □

6.2 Hardness of GenStat with respect to relation $\mathbf{R}_{\text{sim}}$

We defined $\mathbf{R}_{\text{sim}}$ in figure 1. Recall that $\mathbf{R}_{\text{sim}}$ is undecidable in polynomial time in general. But our analysis will ensure that $\mathbf{R}_{\text{sim}}$ is tested only on inputs on which $\Pi^{\mathcal{T}}$ does halt (and in fact halts in a polynomial number of steps). To capture this, we defined a bounded variant of this relation, namely $\mathbf{R}^a_{\text{sim}}$ for every polynomial $a : \mathbb{N} \to \mathbb{N}$.

Relation $\mathbf{R}^a_{\text{sim}}$: Let $a : \mathbb{N} \to \mathbb{N}$ be a polynomial; relation $\mathbf{R}^a_{\text{sim}}$ is identical to $\mathbf{R}_{\text{sim}}$ except that the witness $(u, \Pi^{\langle\cdot\rangle}, \mathcal{T})$ satisfies following additional constraints:
1. $|\mathcal{T}| \leq a(n)$,
2. $\Pi^{\mathcal{T}}$ halts in at most $a(n)$ steps.

Note that $\mathbf{R}^a_{\text{sim}}$ can be tested in time $\mathsf{poly}(a(n)) = \mathsf{poly}(n)$. $\mathbf{L}_{\text{sim}}$ (resp. $\mathbf{L}^a_{\text{sim}}$) is the language corresponding to relation $\mathbf{R}_{\text{sim}}$ (resp., $\mathbf{R}^a_{\text{sim}}$) and $\mathbf{L}^a_{\text{sim}} \in \mathbf{NP}$.

The following lemma states that it is hard for any PPT machine P_1^* to compute a witness ω to statements λ when λ is the transcript of GenStat between P_1^* and an honest V_1. The proof follows [Bar01].

Lemma 1 (Hardness of GenStat). *Assume that $\{\mathcal{H}_n\}$ is a family of collision-resistant hash functions against (non-uniform)* PPT *algorithms. There exists a negligible function* negl *such that for every (non-uniform)* PPT *Turing machine P_1^* , the probability that P_1^*, after interacting with an honest V_1 in protocol* GenStat, *writes a string ω on its (private) output tape such that $\mathbf{R}_{\text{sim}}(\lambda, \omega) = 1$ is at most $\mathsf{negl}(n)$, where λ is the transcript of interaction between P_1^* and V_1, and the probability is taken over the randomness of both P_1^* and V_1.*

Remark. We note that since P_1^* is polynomial time, it can only write a ω of polynomial length. However, since we have to consider all polynomial time P_1^*, it is not known in advance how large ω will be even though it will be of polynomial size.

Proof of lemma 1. Assume, on the contrary, that there exist polynomials p, q and a prover P_1^* such that P_1^* takes at most $p(n)$ steps and writes a string ω on its private output tape such that for infinitely many values of n, $\delta(n) \geq 1/q(n)$ where $\delta(n)$ is the probability that $\mathbf{R}_{\text{sim}}(\lambda, \omega) = 1$ (where λ is sampled as defined in the lemma). Now consider the machine P_1^* in an execution of GenStat and let (h, c) be the first two messages in this interaction. Let the machine $P^*_{1,h,c}$ denote the machine P_1^* whose state has been frozen up to the point where c is sent in this execution. By a standard averaging argument, it follows that with probability at least $\delta/2$ over the sampling of (h, c) in this interaction, the probability that $P^*_{1,h,c}$ writes a valid witness ω at the end of the interaction is at least $\delta/2$. We call such (h, c) "good."

The following procedure finds collisions in h provided (h, c) are good: the procedure chooses two random strings r_1, r_2 each of length n, feeds P_1^* with r_1 and then with r_2 separately; let $\omega_i = (u_i, \Pi_i^{\langle\cdot,\cdot\rangle}, \mathcal{T}_i)$ be the contents of the private output tape of $P^*_{1,h,c}$ when fed with string r_i for $i \in \{1,2\}$. The procedure outputs (Π_1, Π_2) as the potential collision on h.

We claim that the procedure finds collisions in h with noticeable probability as follows. Note that since (h,c) is good, with probability $\delta^2/4$, it holds that $\mathbf{R}_{\text{sim}}(\lambda_i, \omega_i) = 1$ where $\lambda_i = (h, c, r_i)$. Hence, $\Pi_i^{\mathcal{T}_i} = r_i$ and $h(\Pi_1) = h(\Pi_2)$ w.h.p. since c is perfectly binding.

Now, define $\mathcal{I}$ to be an inversion oracle which on input a query of the form $(f, \widetilde{s})$ for $f \in \mathcal{F}_n$ and $\widetilde{s} \in \text{Range}(f)$ outputs $s = f^{-1}(\widetilde{s})$. Then, by definition of $\mathbf{R}_{\text{sim}}$ (in particular, due to condition 4 in figure 1), we have that the output of $\Pi_i^{\mathcal{T}_i}$ is the same as that of $\Pi_i^{\mathcal{I}}$. I.e., $\Pi_i^{\mathcal{I}}$ outputs r_i. Since $\Pi_i^{\mathcal{I}}$ is a *deterministic* computation, it holds that Π_1 and Π_2 are different programs whenever $r_1 \neq r_2$ (which happens with prob. $1 - 2^{-n}$). Further, since P_1^* runs in time at most $p(n)$, programs Π_1, Π_2 are of size at most $p(n)$. Therefore, Π_1 and Π_2 are collisions in h, found with probability at least $\frac{\delta^2}{4} \cdot (1 - 2^{-n}) \geq \delta^2/8$.

It follows that collisions can be found for a noticeable (specifically, at least $\delta/2$) fraction of functions in $\{\mathcal{H}_n\}$ with noticeable probability (specifically, $\delta^2/8$). This concludes the proof. ■

6.3 A Nice Sampler for TM

Protocol GenStat allows us to build a (non uniform) sampling algorithm Samp which will be nice according to definition 6. Samp uses the following simple TM, which was defined earlier:

```
SimLock($\lambda, \omega, s$):
    Test if $\mathbf{R}_{\text{sim}}(\lambda, \omega) = 1$, and if so output $s$;
    Else, output the empty string $0^n$.
```

Also, for a fixed (λ, s), define $\mathsf{SimLock}_{\lambda,s}(\cdot) := \mathsf{SimLock}(\lambda, \cdot, s)$. Machine $\mathsf{SimLock}_{\lambda,s}$ essentially tests whether the input is a valid witness to λ, and if so outputs the fixed value s, and nothing otherwise. Note that it is possible that SimLock takes 2^n steps on some inputs. However, no such inputs will be returned by any PPT adversary who uses (an obfuscation of) SimLock. Also, w.l.o.g., we assume that $\mathsf{Steps}(\mathsf{SimLock}_{\lambda,s_1}, \omega) = \mathsf{Steps}(\mathsf{SimLock}_{\lambda,s_2}, \omega)$ for every λ, ω and $(s_1, s_2) \in \{0,1\}^n \times \{0,1\}^n$.

The sampler. The sampling algorithm, $\mathsf{Samp}_{P_1^*}$ is defined with respect to an arbitrary PPT interactive TM P_1^*. ITM P_1^* follows the instructions of GenStat protocol and interacts with algorithm V_1.

$\mathsf{Samp}_{P_1^*}(1^n; z)$.
- z is of the form $(h, r, s) \in \mathcal{H}_n \times \{0,1\}^n \times \{0,1\}^n$.
- Sample a random transcript λ of GenStat by interacting with P_1^* honestly by sending h as the first message and r as the third message. Let c be the output of P_1^* so that $\lambda = (h, c, r)$.
- Output $\big(z,\ \mathsf{SimLock}_{\lambda,s},\ \mathsf{SimLock}_{\lambda,0^n}\big)$

When the third component of z is fixed to a specific s, we will denote the sampler by Samp_{s,P_1^*} to emphasize a fixed s. The following lemma is essentially a corollary of lemma 1. It proves a stronger claim by directly about Samp_{s,P_1^*}; it is easy to see that the claim will trivially follow for a random s since it follows for each one of them.

Lemma 2. *For every non-uniform* PPT TM P_1^**, and every* $s \in \{0,1\}^n$*,* Samp_{s,P_1^*} *is a nice sampler for Turing machines (according to definition 6).*

Proof. Observe that the pair $(\mathsf{SimLock}_{\lambda,s},\ \mathsf{SimLock}_{\lambda,0^n})$ is *always* a pair of compatible TMs, by definition of SimLock. Now suppose that the second property of definition 6 is not satisfied. Then there exists an A, running in time $a(n)$ for some polynomial a, which outputs an x with noticeable probability such that $\mathsf{SimLock}_{\lambda,s}(x) \neq \mathsf{SimLock}_{\lambda,0^n}(x)$, and $|x| \leq a(n)$; here the probability is taken over the sampling of λ.It follows, from the definition of $\mathsf{SimLock}_{\lambda,s}$, that x must be a witness to λ and therefore A is a PPT machine which finds witnesses to statements $\lambda \in \mathbf{L}_{\text{sim}}^{a}$ with noticeable probability. We can use A to violate lemma 1 as follows.

Consider the machine $B_{1,s}^*$ which incorporates P_1^* and A. It then samples λ by routing messages between P_1^* and an external (honest) V_1, and returns the output of $A\big(z, \mathsf{SimLock}_{\lambda,s}, \mathsf{SimLock}_{\lambda,0^n}\big)$. It is straightforward to see that $B_{1,s}^*$ violates lemma 1 (for every fixed s).