Non-Black-Box Simulation from One-Way Functions And Applications to Resettable Security*

Kai-Min Chung† Rafael Pass‡ Karn Seth§

February 5, 2013

Abstract

The simulation paradigm, introduced by Goldwasser, Micali and Rackoff, is of fundamental importance to modern cryptography. In a breakthrough work from 2001, Barak (FOCS'01) introduced a novel non-black-box simulation technique. This technique enabled the construction of new cryptographic primitives, such as resettably-sound zero-knowledge arguments, that cannot be proven secure using just black-box simulation techniques.

The work of Barak and its follow-ups, however, all require stronger cryptographic hardness assumptions than the minimal assumption of one-way functions: the work of Barak requires the existence of collision-resistant hash functions, and a very recent result by Bitansky and Paneth (FOCS'12) instead requires the existence of an Oblivious Transfer protocol.

In this work, we show how to perform non-black-box simulation assuming just the existence of one-way functions. In particular, we demonstrate the existence of a constant-round resettably-sound zero-knowledge argument based only on the existence of one-way functions. Using this technique, we determine necessary and sufficient assumptions for several other notions of resettable security of zero-knowledge proofs. An additional benefit of our approach is that it seemingly makes practical implementations of non-black-box zero-knowledge viable.

Keywords non-black-box simulations, resettable security, one-way functions, zero-knowledges

*This version is essentially identical to a version from November 5, 2012 except that the January version includes the new section 6.3.

†Cornell University. Email: chung@cs.cornell.edu.

‡Cornell University. Email: rafael@cs.cornell.edu. Work supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF Award CNS-1217821, NSF CAREER Award CCF-0746990, NSF Award CCF-1214844, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

§Cornell University. Email: karn@cs.cornell.edu.

1 Introduction

Zero-knowledge (ZK) interactive protocols [GMR89] are paradoxical constructs that allow one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the Verifier. Beyond being fascinating in their own right, ZK proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks.

The zero-knowledge property is formalized using the so-called *simulation paradigm*: for every malicious verifier V^*, we require the existence of a "simulator" S that, given just the input x, can indistinguishably reproduce the view of V^* in an interaction with the honest prover. (We note that the simulation paradigm extends well beyond the notion of zero-knowledge, and is a crucial component of modern definitions of protocol security.) The most typical way of performing such a simulation is using *black-box simulation* [GO94]: here we exhibit a universal simulator S that, given only black-box access to *any* (efficient) V^*, can reproduce the view of V^* in an interaction with the honest prover. Indeed most zero-knowledge protocols (and more generally protocols for secure computation) are analyzed using black-box simulators. But several limitations of black-box simulators are also known; see e.g. [GK96, CKPR01, BGGL01, PTV08].

In a breakthrough result from 2001, Barak [Bar01] demonstrated a new, powerful *non-black-box* simulation technique, and used this technique to construct a constant-round *public-coin* zero-knowledge argument; by the result of [GK96] such protocols cannot be proved zero-knowledge using just black-box simulation. In the same year, Barak, Goldwasser, Goldreich and Lindell [BGGL01] demonstrated that this non-black-box simulation technique could be used to acheive a *new cryptographic primitive* that cannot be proven secure using black-box simulation, namely *resettably-sound zero-knowledge protocols*. In a resettably-sound zero-knowledge protocol, the soundness property is required to hold even if the malicious prover is allowed to "reset" and "restart" the verifier. This model is particularly relevant for cryptographic protocols being executed on embedded devices, such as smart cards. (Since these devices have neither a built-in power supply, nor a non-volatile rewritable memory, they can be "reset" by simply disconnecting and reconnecting the power supply.) Roughly speaking, the reason why non-black-simulation is cruicial for resettably-sound zero-knowledge protocols is that a black-box simulator has essentially the same "powers" as a malicious resetting prover (i.e., it can only reset and restart the verifier); from this observation it follows that, unless $L \in \mathsf{BPP}$, a "good" simulator can be as a successful cheating prover. Since these results, non-black-box simulation techniques have found applications in various other contexts (see e.g. [BG02, Pas04, PR05, DGS09]).

One important limitation of the non-black-box simulation technique of Barak [Bar01] (also present in its follow-up works) is that the technique requires stronger assumptions than those typically needed for constructing zero-knowledge protocols. In particular, the protocol of Barak (using the refinement in [BG02]) relies on the existence of families of collision-resistant hash functions (CRH), and as a consequence, such hash functions are needed in the above applications too.[1] In contrast, for "plain" zero-knowledge (i.e., without, for instance, resettable soundness) one-way functions are both sufficient and necessary [GMW91, HILL99, OW93], leaving open the following question, which is the focus of this work.

Do one-way functions suffice for performing non-black-box simulation (for primitives

[1] The original protocol of Barak relies on a very slightly super-polynomially hard collision-resistant hash function; the need for super-polynomial hardness was removed in [BG02]

that cannot be proven secure using black-box simulation techniques)?

A very recent elegant work by Bitansky and Paneth [BP12a] takes us a step closer to answering this question. They present a resettably-sound zero-knowledge argument without relying on hash functions; instead, they rely on the existence of an *oblivious transfer (OT) protocol*. Although, the existence of an OT protocol is seemingly a more "complex" assumption than the existence of CRHs,[2] it is not known whether the existence of an OT protocol implies the existence of CRH (or vice versa). More important, to achieve this result, Bitansky and Paneth devise a quite different method for performing non-black-box simulation.

1.1 Our Result

In this work, we answer the above question in the affirmative. We show that for the case of resettably-sound zero-knowledge, the existence of one-way functions suffices.

Theorem 1 (Main Theorem)**.** *Assume the existence of one-way function. Then there exists a constant-round resettably-sound zero-knowledge argument for all of* NP*.*

Interestingly, our protocol is quite close in spirit to Barak's original protocol, while dispensing of the need for collision-resistant hash functions.

By relying on the above main theorem, we establish several other results on resettable security: Assuming one-way functions, all of NP has

- a constant-round resettably-witness-indistinguishable argument of knowledge;
- a $\tilde{O}(\log n)$-round resettable-zero-knowledge argument of knowledge.

(Roughly speaking, in a resettably-witness indistinguishable (resp., zero-knowledge) argument, the witness-indistinguishability (resp., zero-knowledge) property is required to hold also in the presence of a resetting verifier. For the above-mentioned primitives, previous results required additional cryptographic assumptions (the existence of collision-resistant hash-functions or oblivious transfer protocols). We additionally show how to eliminate the needs for CRHs in the construction of [DGS09] of a simultaneously resettable zero-knowledge argument for NP—simultaneous resettability here means that security (both zero-knowledge and soundness) holds even with respect to resetting attackers. (Combined with the very recent result of [OV12, CP13], this yields a construction of a simultanously resettable zero-knowledge argument based on one-way functions.)

We emphasize that for all the above results, the use of non-black-box techniques are inherent. Our results lead to improvements also for cases when black-box simulation can be used: prior to our results, resettable zero-knowledge arguments (without the argument of knowledge property) were known only based on the existence of CRHs, but these protocols were actually proven secure using black-box simulation. As mentioned above, we are able to establish even the stronger notion of a resettable zero-knowledge argument *of knowledge* assuming only one-way functions.

[2] For instance, all candidate constructions of OT protocols rely on "structured", number-theoretic or lattice-based, assumptions. Additionally, all these assumptions are known to imply also the existence of collision-resistance hash function (but the converse is not true).

1.2 Our Techniques

To explain our techniques, let us start by very briefly recalling the idea behind Barak's constant-round public-coin protocol; we will then explain how this protocol is used to get a resettably-sound zero-knowledge protocol. The protocol relies on the existence of a family of collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^n$; note that any such family of collision-resistant hash functions can be implemented from a family of collision-resistant hash functions mapping n-bit string into $n/2$-bit strings using *tree hashing* [Mer89].

Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\text{poly}(n)}$, the Prover P and Verifier V, proceed in two stages. In Stage 1, V starts by selecting a function h from a family of collision-resistant hash function and sends it to P; P next sends a commitment $c = \mathsf{Com}(0^n)$ of length n, and finally, V next sends a "challenge" $r \in \{0,1\}^{2n}$. In Stage 2, P shows (using a witness indistinguishable argument of knowledge) that either x is true, or that c is a commitment to a "hash" (using h) of a program M (i.e., $c = \mathsf{Com}(h(M))$ such that $M(c) = r$.

Roughly speaking, soundness follows from the fact that even if a malicious prover P^* tries to commit to (the hash of) some program M (instead of committing to 0^n), with high probability, the a string r sent by V will be different from $M(c)$ (since r is chosen independently of c). To prove ZK, consider the non-black-box simulator S that commits to a hash of the code of the malicious verifier V^*; note that, by definition, it thus holds that $M(c) = r$, and the simulator can use c as a "fake" witness in the final proof. To formalize this approach, the witness indistinguishable argument in Stage 2 must actually be a witness indistinguishable *universal argument* (WIUARG) [Mic00, BG02] since the statement that c is a commitment to a program M of *arbitrary* polynomial-size, and that proving $M(c) = r$ within some *arbitrary* polynomial time, is not in NP. WIUARG are known based on the existence of CRH and those protocols are constant-round public-coin; as a result, the whole protocol is constant-round and public-coin.

Finally, Barak et al. [BGGL01] show that any constant-round public-coin zero-knowledge argument of knowledge can be transformed into a resettable-sound zero-knowledge argument, by simply having the verifier generate its (random) message by applying a pseudorandom function to the current partial transcript.[3]

Why hash functions are needed Note that hash functions are needed in two locations in Barak's protocol. First, since there is no *a-priori* polynomial upper-bound of the length of the code of V^*, we require the simulator to commit to the hash of the code of V^* . Secondly, since there is no *a-priori* polynomial upper-bound on the running-time of V^*, we require the use of universal arguments (and such constructions are only known based on the existence of collision-resistant hash functions).

Using signature schemes instead of CRHs Our main idea is noticing that digital signature schemes—which can be constructed based on one-way functions—share many of the desirable properties of CRHs. In particular, we will show how to appropriately instantiate (a variant of) Barak's protocol using signature schemes instead of using CRHs. Recall that "fixed-length" signature schemes, that allow signing messages of arbitrary polynomial-length (e.g length $2n$) using a length n signature, are known based on just one-way functions [Rom90]. In fact, based on the same assumption, *strong* fixed-length signature schemes are known: in a strong signature scheme no polynomial time attacker can obtain a *new* signature even for messages that it has seen a signature on [Gol01]. We observe that such signature scheme share a lot of properties with CRHs. First of

[3] Strictly speaking, Barak's protocol is not a argument of knowledge, but rather a "weak" argument of knowledge (see [BG02, BGGL01] for more details), but the transformation of [BGGL01] applies also to such protocol.

all, they are compressing. More importantly, we observe that by the unforgeability requirement of strong signatures, no attacker can find a single valid signature σ for two distinct messages m, m'—that is, signatures satisfy a collision-resistance property. Additionally, by using an appropriate analog of tree hashing, a *signature tree* could be used to compress arbitrary length messages into a signature of length n.

So, can we just replace the CRHs in Barak's protocol with strong, fixed-length, signature schemes? The problem with naively implementing this idea is that the collision-resistance property of strong signature schemes only holds against an attacker that does *not* know the secret key. On the other hand, to generate signatures, knowledge of the secret key is needed. In our application, the simulator—acting as a prover—needs to be able to generate signature (in order to "hash down" the program, and in the universal argument) but at the same time, we need to ensure collision-resistance against cheating provers. So if we let the prover generate the signature keys, simulation is easy, but soundness no longer holds, whereas if we let the verifier generate the signature keys and only sends the verification key to the prover, then soundness holds, but it is no longer clear how to perform a simulation. We resolve this issue by using a "hybrid approach": we let the verifier generate the signature keys, but gives the prover access to a *single* signing query. More precisely, in an initial stage of the protocol, the verifier generates a signature key-pair sk, vk and send only the verification key vk to the prover. Next, in a "signature slot", the prover sends a message m to the verifier, and the verifier computes and returns a valid signature σ of m (using sk). (We note that such a signature slot previously used by [LP11] in a quite different context, but as we shall see shortly, some of their techniques will be useful also to us.) Finally, the protocol proceeds essentially as in Barak's protocol, but where the CRH is replaced using the signature scheme. Implementing this is somewhat subtle: First, the statement proved in the WIUARG in Barak's protocol considers the hash function h (e.g., prover needs to prove statements of the type $h(m) = q$). In our approach since "hashing" has been replaced by "signing", this would require the honest prover to prove things related to the secret-key (e.g., $\mathsf{Sign}_{\mathsf{sk}}(m) = q$), but the honest prover does not know sk. This issue is easily resolved by instead of letting the prover show that signatures used (as "hashes") verify—i.e., that $\mathsf{Ver}_{\mathsf{vk}}(m) = q$. Another issue is that in Barak's protocol, the honest prover actually needs to perfom hashes to complete the WIUARG. We resolve this second issue by relying on an instantiation of Barak's protocol due to Pass and Rosen [PR05], which relies on a special-purpose WIUARG, in which the honest prover never needs to perform any hashing.[4] Now completeness of this protocol follows in exactly the same way as in [Bar01, PR05].

For soundness, note that since the prover does not get to see sk, soundness follows in a similar way to Barak's protocol. In fact, if the signature scheme used satisfies strong unforgeability, then the signature trees are collision-resistant with respect to attackers that get vk and *have access to a signing oracle*, and collision-resistance of the signature tree is the only property needed to prove soundness as in Barak's protocol. (Note that we here only require collision-resistance with respect to attackers that get a single query to a signing oracle, but the more general result will be useful when we consider resettable-soundness.)

Let us turn to zero-knowledge. At first sight, it seems that we still have an issue. The prover just gets a single signature, but to complete the simulation, the simulator needs an a-priori unbounded polynomial number of signatures (to e.g., "hash down" a program of a-priori unbounded polynomial-size.[5]) Note, however, that the simulator can always *rewind* the verifier to get as many signatures as

[4] In fact, an early version of Barak's protocol also had this property.

[5] Also in the implementation of the WIUARG, an a-priori unbounded number of "hashes" are needed.

it wants and can thus complete the simulation in a similar way to the one used in Barak's protocol. This approach doesn't quite work: the malicious verifier V^* may not always agree to sign every message requested by the simulator; we deal with this issue in the same way as in [LP11], rather than having the simulator send the messages it wants to be signed in the clear, it simply sends a commitment to them. To make use of such a simulator strategy, we appropriately modify the notion of a signature tree to consist of signatures *of commitments* to signatures etc; we refer to this type of a signature tree as a "sig-com" tree.

So, we now have a zero-knowledge protocol that is based on one-way functions (and is constant-round). But it is no longer public-coin!

Nonetheless, let us still apply the PRF transformation of [BGGL01] to the protocol (i.e., we have the verifier generate its random coins in each round by applying a PRF to the current partial transcript). Clearly, the protocol is still zero-knowledge (since we only modified the verifier strategy). As it turns out, the resulting protocol is actually also resettably-sound: note that, except for the signature slot added in the beginning of the protocol, the protocol still is public-coin, and the same argument as in [GK96, BGGL01] can be used to show that in the public-coin part of the protocol, rewindings do not "help" a resetting cheating prover. So, in essence, the only "advantages" a resetting prover gets is that it may rewind the signature slot, and thus get an arbitrary polynomial number of signatures on messages of its choice. But, as noted above, signature trees are collision-resistant even with respect to an attacker that gets an arbitrary polynomial number of queries to a signing oracle and thus resettable-soundness follows in exactly the same way as the (non-resetting) soundness property.

Beyond resettably-sound zero-knowledge For the applications of a) a constant-round resettably witness-indistinguishable argument of knowledge, and b) $\tilde{O}(\log n)$-round resettable-zero-knowledge argument of knowledge for NP, we simply plug in our resettably-sound zero-knowledge argument of knowledge into the protocols of [CGGM00, BGGL01] with some minor modifications.

To achieve simulateously resettable zero-knowledge, we instead instantiate the protocol of Deng, Goyal and Sahai [DGS09] with signature trees, in exactly the same way as Barak's protocol. Resettable-soundness follows exactly as in [DGS09], relying on the collision-resistance property of signature trees. Resettable-zero-knowledge is more tricky though: [DGS09] provides an intricate simulation strategy that combines black-box simulation, using rewinding, and non-black-box simulation (as in [Bar01]). Roughly speaking, the protocol consists of polynomially many "rewinding slots" (say $2n^2$), and for each session started by the resetting verifier, the simulator of [DGS09] rewinds a polynomial fraction (say $2n$) of them *twice*. Their argument shows that for each such slot, the rewinding "succeeds" with probability close to $1/2$ and the slot gets "solved"; as a consequence, except with negligible probability, for each session, there exists some slot that is "solved" and this suffices for simulating the session. In our instantiation of their protocol, rewinding a slot just once does not suffice to "solve" the session (and complete the simulation of that session). Rather we need polynomially many, say $g(n) = \text{poly}(|V^*|)$ where $|V^*|$ is the size of the verifier (including its auxiliary input), successful rewindings (in order to rewind the signature slot sufficiently many times to provide the signature trees). We deal with this issue in a straight-forward way: we use exactly the same rewinding strategy as in [DGS09] but instead rewind each slot (that was being rewound once in [DGS09]) $3g(n)$ times. It follows using a slight generalization of the argument in [DGS09] that each slot that is rewound is successfully solved with probability close to $1/2$, and the rest of the simulation argument continues in identically the same way as [DGS09]. Additionally, rewinding polynomially many times (as opposed to twice) only increases the running-time by a polynomial

factor (the technical reason for this is that the [DGS09] simulator only performs a constant-number of recursive rewindings).

A PCP-free construction Just as the construction of Barak's protocol, our constructions rely on universal arguments, which in turn rely on Probabilistically Checkable Proofs (PCPs). Intriguingly, the approach of Bitansky and Paneth [BP12a] does not rely on PCPs; on the other hand, it relies on some other quite heavy machinery: "unobfuscatable functions" [BGI+12] and general secure two-party computation [GMW91].

As we now sketch, our approach can be instantiated without the use of PCPs, and without introducing any other machinery. (Indeed, although we have not verified the details, it would seem that a practical implementation of our protocol can be given by relying on efficient signatures and zero-knowledge proofs of committed signatures, as in e.g., [CL01].) Recall that in Barak's protocol the universal argument is used to prove a statement of the form c is a commitment to a hash of a program M such that $M(c) = r$. Also recall that (in the [PR05] variant of [Bar01]) the honest prover never needs to engage in the universal argument, it is only the simulator that needs to prove the above statement. Rather than providing a universal argument, we let the simulator prove this statement *piecemeal*, step-by-step, using a strategy that is very similar to one employed in the "impossibility of instantiating random oracles" result of [CGH04][6] (On a high-level, this type of piecemeal decomposition is also somewhat similar to what is done in the impossibility result of [BGI+12]; as such our approach brings out the connection between the techniques from [Bar01] and [BP12a].) More precisely, in the actual protocol, the verifier generates a key-pair $\mathsf{vk}', \mathsf{sk}'$ for a signature scheme and sends vk' to the prover. The prover then provides the verifier with a commitment c_1 to a three hash[7] of a *start-configuration*, a commitment c_2 to a tree-hash of a *current-configuration*, a commitment c_3 to a tree-hash of a *next-configuration*, a commitment c_4, and a witness indistinguishable argument of knowledge that either a) $x \in L$ or b) *start-configuration* = *next-configuration* or c) c_4 is a commitment to a signature (using sk') of c_1, c_2 and if performing *one step* of computation given *current-configuration* leads to *next-configuration*. (Note that since we use tree-hashing, verification of condition b) and c) can both be done in time polylogarithmic in the length of the configurations). If the argument of knowledge is accepting, the verifier signs c_1, c_3. Roughly speaking, the above "slot" makes it possible for the simulator to get a signature on (commitments to signature-trees) (s_0, s_0) where s_0 is the initial configuration of $M(\sigma)$ (using condition b), and next by rewinding the verifier sufficiently many times to get signatures on later configurations (s_0, s_t) in the computation of $M(\sigma)$ (using condition c). Thus, finally, the simulator can get a signature on (s_0, s_T) where s_T is the terminating configuration of the computation of $M(\sigma)$. The simulator can then use this signature to convince the verifier that $M(c) = r$ where M is the program committed to in c. A complete formalization of this approach will appear in the final version of this paper.

1.3 Subsequent Work

A very recent elegant work by Bitansky and Paneth [BP12b] (developed subsequently to our results) shows an alternative approach for obtaining resettably-sound arguments (and related primitives)

[6] They key difference is that construction of [CGH04] only considers an honest "non-aborting" verifier, whereas we need to deal with also malicious "aborting" verifiers. This issue is analogous to why we rely on "sig-com" trees (consisting of signatures *of commitments* to signatures etc.) as opposed to "plain" signature trees.

[7] We may also instantiate tree-hashing with signature-trees to get an implementation based on one-way functions.

from one-way functions, by first constructing functions that not even are "approximately" unobfuscatable, and relying on the connection between resettable-soundness and unobfuscatable functions from [BP12a].

Following the recent works of [OV12, CP13] ([OV12] was developed independently of the current work, and [CP13] subsequently to both [OV12] and the current work), we have added Section 6.2, demonstrating how to get simultaneously resettable zero-knowledge from only one-way functions. Independently, [BP12b] have also established the same results by instantiating their framwork using the result from [CP13].

1.4 Outline

In Section 3 we provide formal definitions of signature trees, and provide collision-resistance properties of such trees. To formalize our construction of resettably-sound zero-knowledge in a modular way, in Section 4, we first consider an "oracle-aided" model, in which players have access to a signing oracle. We first show that the universal argument construction of Barak and Goldreich [BG02] can be instantiated using one-way functions in such an oracle-aided model, by replacing "hashing" with "signing". We next show how to instantiate Pass and Rosen's [PR05] variant of Barak protocol in the same way (by relying on the oracle-aided construction of universal arguments). This leads to a constant-round oracle-aided public-coin zero-knowledge argument of knowledge, satifying a key property: the honest prover never needs to access the oracle. We may next apply the transformation of [BGGL01] to this protocol to obtain an oracle-aided resettably-sound zero-knowledge argument of knowledge satisfying the same key property (the results of [BGGL01] relativize and thus we can directly apply them also to oracle-aided protocols).

In Section 5, we present a general transformation, transforming any oracle-aided resettably-sound zero-knowledge argument (of knowledge) satisyfing the above key property, into a resettably-sound zero-knowledge argument (of knowledge) in the "plain" model (i.e. without any oracle): the transformation simply consists of adding a signature slot at the beginning of the protocol. Taken together with our result in Section 4, this yields a constant-round resettably-sound zero-knowledge argument of knowledge for NP based on one-way functions.

In Section 6, applications (such as simultanously resettable zero-knowledge) are presented.

2 Preliminaries

We assume familiarity with interactive arguments, arguments of knowledge and witness indistinguishability; see the Appendix A for more details.

2.1 Notations

Let $\mathbb{N}$ denote the set of positive integers, and $[n]$ denote the set $\{1, 2, \ldots, n\}$. Given a string x, we let x_i denote the i-th bit of x, and $x_{\leq i}$ denote the prefix of x upto and including its ith bit. By a probabilistic algorithm we mean a Turing machine that receives an auxiliary random tape as input. If M is a probabilistic algorithm, then for any input x, the notation "$M_r(x)$" denotes the output of the M on input x when M's random tape is fixed to r, while $M(x)$ represents the distribution of outputs of $M_r(x)$ when r is chosen uniformly. An oracle algorithm is a machine that gets oracle access to another machine. Given a probabilistic oracle algorithm M and a probabilistic algorithm

A, we let $M^A(x)$ denote the probability distribution over the outputs of the oracle algorithm M on input x, when given oracle access to A.

By $x \leftarrow \mathcal{S}$, we denote an element x is sampled from a distribution $\mathcal{S}$. If F is a finite set, then $x \leftarrow F$ means x is sampled uniformly from the set F . To denote the ordered sequence in which the experiments happen we use semicolon, e.g. $(x \leftarrow \mathcal{S}; (y,z) \leftarrow A(x))$. Using this notation we can describe probability of events. For example, if $p(\cdot,\cdot)$ denotes a predicate, then $Pr[x \leftarrow \mathcal{S}; (y,z) \leftarrow A(x) : p(y,z)]$ is the probability that the predicate $p(y,z)$ is true in the ordered sequence of experiments $(x \leftarrow \mathcal{S}; (y,z) \leftarrow A(x))$. The notation $\{(x \leftarrow \mathcal{S}; (y,z) \leftarrow A(x) : (y,z))\}$ denotes the resulting probability distribution $\{(y,z)\}$ generated by the ordered sequence of experiments $(x \leftarrow \mathcal{S}; (y,z) \leftarrow A(x))$.

2.2 Zero Knowledge

We start by recalling the definition of zero knowledge from [GMR89].

Definition 1 (Zero-knowledge [GMR89])**.** *An interactive protocol (P,V) for language L is **zero-knowledge** if for every PPT adversarial verifier V^*, there exists a PPT simulator S such that the following ensembles are computationally indistinguishable over $x \in L$:*

$$\{\mathsf{View}_{V^*}\langle P, V^*(z)\rangle(x)\}_{x\in L, z\in\{0,1\}^*} \approx \{S(x,z)\}_{x\in L, z\in\{0,1\}^*}$$

2.3 Resettably Sound Zero Knowledge

Let us recall the definition of resettable soundness due to [BGGL01].

Definition 2 (Resettably-sound Arguments [BGGL01])**.** *A resetting attack of a cheating prover P^* on a resettable verifier V is defined by the following two-step random process, indexed by a security parameter n.*

1. *Uniformly select and fix $t = poly(n)$ random-tapes, denoted $r_1, \ldots, r_t$, for V, resulting in deterministic strategies $V^{(j)}(x) = V_{x,r_j}$ defined by $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$,[8] where $x \in \{0,1\}^n$ and $j \in [t]$. Each $V^{(j)}(x)$ is called an incarnation of V .*

2. *On input 1^n, machine P^* is allowed to initiate $poly(n)$-many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0,1\}^n$ and $j \in [t]$, thus defining $V^{(j)}(x)$, and conducts a complete session with it.*

Let (P,V) be an interactive argument for a language L. We say that (P,V) is a resettably-sound argument *for L if the following condition holds:*

- Resettable-soundness*: For every polynomial-size resetting attack, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L$ is negligible.*

We will also consider a slight weakening of the notion of resettable soundness, where the statement to be proven is fixed, and the verifier uses a single random tape (that is, the prover cannot start many independent instances of the verifier).

[8] Here, $V(x, r, \alpha)$ denotes the message sent by the strategy V on common input x, random-tape r, after seeing the message-sequence α.

Definition 3 (Fixed-input Resettably-sound Arguments [PTW09]). *An interactive argument (P, V) for a NP language L with witness relation R_L is* ***fixed-input resettably-sound*** *if it satisfies the following property: For all non-uniform polynomial-time adversarial prover P^*, there exists a negligible function $\mu(\cdot)$ such that for every all $x \notin L$,*

$$\Pr[R \leftarrow \{0,1\}^\infty; (P^{*V_R(x,\mathsf{pp})}, V_R)(x) = 1] \leq \mu(|x|)$$

As the following claim shows, any zero-knowledge *argument of knowledge* satisfying the weaker notion can be transformed into one that satisfies the stronger one, while preserving zero-knowledge (or any other secrecy property against malicious verifiers.

Claim 2. *Let (P, V) be a fixed-input resettably sound zero-knowledge (resp. witness indistinguishable) argument of knowledge for a language $L \in NP$. Then there exists a protocol (P', V') that is a (full-fledged) resettably-sound zero-knowledge (resp. witness indistinguishable) argument of knowledge for L.*

Proof. (Sketch): We rely on the PRF transformation used in [BGGL01], but since we are dealing with private-coin protocols, we simply apply it to the random tape of the verifier. More precisely, the new verifier V' now chooses its random coins by applying a PRF to the statement x, and then continues its execution by simulating V using these random coins. (The honest prover remains unchanged). It follows using identically the same proof as in [BGGL01] that the new protocol satisfies the require properties (note that the argument of knowledge property is necessary in this argument). Since we have only modified the verifier strategy, zero-knowledge (or witness indistinguishability) still holds. □

3 Signature Trees

In this section, we define an analogue of Merkle-hash trees using signature schemes. Towards this, we will rely on the existence of strong, fixed-length, deterministic secure signature schemes. Recall that in a strong signature scheme, no polynomial-time attacker having oracle access to a signing oracle can produce a valid message-signature pair, unless it has received this pair from the signing oracle. The signature scheme being fixed-length means that signatures of arbitrary (polynomial-length) messages are of some fixed polynomial length. Deterministic signatures don't use any randomness in the signing process once the signing key has been chosen. In particular, once a signing key has been chosen, a message m will always be signed the same way.

Definition 4 (Strong Signatures). *A* strong, length-ℓ, signature scheme SIG *is a triple* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *of* PPT *algorithms, such that*

1. *for all $n \in \mathbb{N}, m \in \{0,1\}^*$,*

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), \sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m); \mathsf{Ver}_{\mathsf{vk}}(m, \sigma) = 1 \wedge |\sigma| = \ell(n)] = 1$$

2. *for every non-uniform* PPT *adversary A, there exists a negligible function $\mu(\cdot)$ such that*

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), (m, \sigma) \leftarrow A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^n); \mathsf{Ver}_{\mathsf{vk}}(m, \sigma) = 1 \wedge (m, \sigma) \notin L] \leq \mu(n),$$

where L denotes the list of query-answer pairs of A's queries to its oracle.

Strong, length-ℓ, deterministic signature schemes with $\ell(n) = n$ are known based on the existence of OWFs; see [NY89, Rom90, Gol01] for further details. In the rest of this paper, whenever we refer to signature schemes, we always means strong, length-n signature schemes.

Let us first note that signatures satisfy a "collision-resistance" property.

Claim 3. *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong (length-n) signature scheme. Then, for all non-uniform PPT adversaries A, there exists a negligible function* $\mu(\cdot)$ *such that for every* $n \in \mathbb{N}$,

$$Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), (m_1, m_2, \sigma) \leftarrow A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^n, \mathsf{vk}); \mathsf{Ver}_{\mathsf{vk}}(m_1, \sigma) = \mathsf{Ver}_{\mathsf{vk}}(m_2, \sigma) = 1] \leq \mu(n)$$

Proof. Assume for contradiction that there exists some non-uniform polynomial-time A such that A breaks "collision-resistance" property of SIG with probability $\frac{1}{p(n)}$ for infinitely many $n \in \mathbb{N}$, where p is a polynomial. We show that A can be used to break the strong unforgeability property of SIG. More precisely, note that if A outputs a valid signatures $(m_1, \sigma), (m_2, \sigma)$ without querying the signing oracle with m_1 and m_2 and receiving σ as a response to both queries, then A already breaks the security of the signature scheme. Thus w.l.o.g. we may assume A queries both m_1 and m_2 to the signing oracle and receives σ as a response. We then simulate A, recording the previous messages queried to the oracle along with the responses. At each point during the execution of A, before forwarding the next query m to the oracle, we test if any of the previously received signatures are valid signatures for m. If so, we output m together with such a signature σ. Notice that if A always queries m_1 and m_2 and receives σ as a response, then we will intercept whichever of the two A queries second. Thus, for infinitely many n, with probability $\geq \frac{1}{p(n)}$, we forge a signature σ for some m before ever querying the signing oracle and receiving σ as a response. □

We now define an analog of Merkle-hash tree which we call *signature trees* and show that they also satisify a collision-resistant property. We index each node of a complete binary tree Γ of depth d by a binary string of length at most d, where the root is indexed by the empty string λ, and each node indexed by γ has left and right children indexed $\gamma 0$ and $\gamma 1$, respectively.

Definition 5 (Signature Trees)**.** *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n signature scheme. Let* $(\mathsf{sk}, \mathsf{vk})$ *be a key-pair of* SIG*, and s be a string of length* 2^d*. A* signature tree *of the string s w.r.t.* $(\mathsf{sk}, \mathsf{vk})$ *is a complete binary tree of depth d, defined as follows.*

- *A leaf* l_γ *indexed by* $\gamma \in \{0,1\}^d$ *is set as the bit at position* γ *in s.*
- *An internal node* l_γ *indexed by* $\gamma \in \bigcup_{i=0}^{d-1}\{0,1\}^i$ *satisfies that* $\mathsf{Ver}_{\mathsf{vk}}((l_{\gamma 0}, l_{\gamma 1}), l_\gamma) = 1$.

Note that to *verify* whether a Γ is a valid signature-tree of a string s w.r.t. the signature scheme SIG and the key-pair $(\mathsf{sk}, \mathsf{vk})$ knowledge of the secret key sk is not needed. However, to *create* a signature-tree for a string s, the secret key sk is needed.

The following notion of a signature path is the natural analog of an authentication path in a Merkle-tree.

Definition 6 (Signature Path)**.** *A* signature path *w.r.t.* SIG*,* vk *and a root* l_λ *for a bit b at leaf* $\gamma \in \{0,1\}^d$ *is a vector* $\vec{\rho} = ((l_0, l_1), ((l_{\gamma_{\leq 1}0}, l_{\gamma_{\leq 1}1}), \ldots (l_{\gamma_{\leq d-1}0}, l_{\gamma_{\leq d-1}1}))$ *such that for every* $i \in \{0, \ldots, d-1\}$, $\mathsf{Ver}_{\mathsf{vk}}((l_{\gamma_{\leq i}0}, l_{\gamma_{\leq i}1}), l_{\gamma_{\leq i}}) = 1$. *Let* $\mathsf{PATH}^{\mathsf{SIG}}(\vec{\rho}, b, \gamma, l_\lambda, \mathsf{vk}) = 1$ *if* ρ *is a signature path w.r.t.* SIG*,* vk*,* l_λ *for b at* γ.

The following claim states that signature trees also satisfy an appropriate collision-resistance property: no non-uniform PPT attacker having oracle access to a signing oracle can output a root and valid signature paths for both 0 and 1 at some leaf γ.

Claim 4. *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n, signature scheme. Then, for every non-uniform* PPT *adversary* A*, there exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*,*

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), (\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^n, \mathsf{vk}); \\ \forall b \in \{0,1\} \ \ \mathsf{PATH}^{\mathsf{SIG}}(\vec{\rho}_b, b, \gamma, l_\lambda, vk) = 1] \leq \mu(n)$$

Proof. The claim directly follows from Claim 3 since any two valid signature-paths with the same root but different leaf value must contain a collision for the underlying signature scheme. □

3.1 Sig-Com Schemes

For the technical reason explained in the introduction, we will rely on variant of signature trees consisting of alternating signatures and commitments. To formalize this, we consider the notion of a "sig-com" scheme:

Definition 7 (Sig-Com Schemes)**.** *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n, signature scheme, and let* Com *be a non-interactive commitment schemes. Define* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *to be a triple of* PPT *machines defined as follows:*

- $\mathsf{Gen}' = \mathsf{Gen}$.
- $\mathsf{Sign}'_{\mathsf{sk}}(m)$: *compute a commitment* $c = \mathsf{Com}(m; \tau)$ *using a uniformly selected* τ*, and let* $\sigma = \mathsf{Sign}_{\mathsf{sk}}(c)$*; output* (σ, τ)
- $\mathsf{Ver}'_{\mathsf{vk}}(m, \sigma, \tau)$: *Output* 1 *iff* $\mathsf{Ver}_{\mathsf{vk}}(\mathsf{Com}(m, \tau), \sigma) = 1$.

We call SIG' *the* Sig-Com Scheme *corresponding to* SIG *and* Com.

Note that the above definition of a sig-com scheme assumes that Com is a non-interactive commitment scheme. This is only for convenience of notation; the above definition, as well as all subsequent results directly apply also to 2-round commitment (i.e., families of non-interactive commitment schemes, as in [Nao91]), by simply adding the first message to the verification key of the sig-com scheme.

Sig-com schemes also satisfy a collision-resistant property:

Claim 5 (Collision Resistance of Sig-Coms)**.** *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n signature scheme,* Com *be non-interactive commitment scheme, and let* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *be the sig-com scheme corresponding to* SIG *and* Com*. Then, for any non-uniform* PPT *adversary* A*, there exists a negligible function* μ *such that for all* $n \in \mathbb{N}$*:*

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), (\sigma, m_1, m_2, \tau_1, \tau_2) \leftarrow A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^n, \mathsf{vk}); \\ m_1 \neq m_2, \mathsf{Ver}'_{\mathsf{vk}}(m_1, \sigma, \tau_1) = \mathsf{Ver}'_{\mathsf{vk}}(m_2, \sigma, \tau_2) = 1] \leq \mu(n)$$

Proof. Note that by the binding property of Com, no non-uniform PPT can output a valid commitment c to two different messages $m_1 \neq m_2$ except with negligible probability. Thus, except with negligible probability, a successful non-uniform PPT attacker must output a signature for two different commitments $c_1 \neq c_2$, violating collision-resistance of SIG (i.e., Claim 3). □

Note that in Claim 5, the attacker gets oracle access to a signature oracle (for SIG) as opposed to a sig-com oracle. We may now define sig-com trees and sig-com paths in an analogous way to (plain) signature trees and paths.

Definition 8 (Sig-Com Trees)**.** *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n signature scheme, let* Com *be a non-interactive commitment and let* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *be the sig-com scheme corresponding to* SIG *and* Com*. Let* $(\mathsf{sk}, \mathsf{vk})$ *be a key-pair of* SIG'*, and* s *be a string of length* 2^d*. A* signature tree *of the string* s *w.r.t.* $(\mathsf{sk}, \mathsf{vk})$ *is a complete binary tree of depth* d*, defined as follows.*

- *A leaf* l_γ *indexed by* $\gamma \in \{0,1\}^d$ *is set as the bit at position* γ *in* s*.*
- *An internal node* l_γ *indexed by* $\gamma \in \bigcup_{i=0}^{d-1}\{0,1\}^i$ *satisfies that there exists some* τ_γ *such that* $\mathsf{Ver}'_{\mathsf{vk}}((l_{\gamma 0}, l_{\gamma 1}), l_\gamma, \tau_\gamma) = 1$*.*

Definition 9 (Sig-Com Path)**.** *Let* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *be a sig-com scheme. A* sig-com path *w.r.t.* SIG'*,* vk *and a root* l_λ *for a bit* b *at leaf* $\gamma \in \{0,1\}^d$ *is a vector* $\vec{\rho} = ((l_0, l_1, \tau_\lambda), ((l_{\gamma_{\leq 1}0}, l_{\gamma_{\leq 1}1}, \tau_{\gamma_{\leq 1}}), \ldots, (l_{\gamma_{\leq d-1}0}, l_{\gamma_{\leq d-1}0}, \tau_{\gamma_{\leq d-1}})$ *such that for every* $i \in \{0, \ldots, d-1\}$*,* $\mathsf{Ver}'_{\mathsf{vk}}((l_{\gamma_{\leq i}0}, l_{\gamma_{\leq i}1}), l_{\gamma_{\leq i}}, \tau_{\gamma_{\leq i}})) = 1$*. Let* $\mathsf{PATH}^{\mathsf{SIG}'}(\vec{\rho}, b, \gamma, l_\lambda, \mathsf{vk}) = 1$ *if* $\vec{\rho}$ *is a signature path w.r.t.* SIG'*,* vk*,* l_λ *for* b *at* γ*.*

Sig-com trees also satisfy a collision-resistance property:

Claim 6. *Let* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *be a strong, length-n signature scheme, let* Com *be a non-interactive commitment and let* $\mathsf{SIG}' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Ver}')$ *be the sig-com scheme corresponding to* SIG *and* Com*. Then, for every non-uniform* PPT *adversary* A*, there exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*,*

$$\Pr[(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n), (\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow A^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(1^n, \mathsf{vk}); \\ \forall b \in \{0,1\} \;\; \mathsf{PATH}^{\mathsf{SIG}'}(\vec{\rho}_b, b, \gamma, l_\lambda, \mathsf{vk}) = 1] \leq \mu(n)$$

Proof. As in Claim 4, the claim follows directly from Claim 5 since any two valid sig-com paths with the same root but different leaf values must contain a collision for the underlying sig-com scheme. □

Canonical Sig-com Schemes Throughout the rest of the paper, we consider sig-com schemes SIG' and sig-com trees corresponding to a strong, length-n deterministic signature scheme SIG and a non-interactive commitment Com that generates n^2 bits long commitments to $2n$ bits strings. Thus, each node of the sig-com tree is an n-bit signature of an n^2 bits commitment of the two signatures of the children nodes. Hereafter, we refer to such a SIG' as a **canonical sig-com scheme**.

4 Oracle-Aided Resettably-sound Zero Knowledge Protocols

In this section we show how to construct a resettably-sound ZK argument in an oracle-aided model where prover and verifier additionally have access to a public parameter generated prior to the interaction (in our protocol, this will be the verification key for a signature scheme), and, further the prover has access to an oracle, also generated prior to the interaction (in our protocol, this will be a signature/sig-com oracle).

More formally, let $\mathcal{O}$ be a probabilistic algorithm that on input a security parameter n, outputs a polynomial-length (in n) public-parameter pp, as well as the description of an oracle O. The oracle-aided execution of an interactive protocol with common input x between a prover P with auxiliary input y and a verifier V consist of first generating $\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|x|})$ and then letting $P^O(x, y, \mathsf{pp})$ interact with $V(x, \mathsf{pp})$.

Definition 10 (Oracle-aided Interactive Arguments)**.** *A pair of oracle algorithms (P, V) is an $\mathcal{O}$-**oracle aided argument** for a NP language L with witness relation R_L if it satisfies the following properties:*

- *Completeness: There exists a negligible function $\mu(\cdot)$, such that for all $x \in L$, if $w \in R_L(x)$,*

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|x|}); (P^O(w), V)(x, \mathsf{pp}) = 1] = 1 - \mu(|x|)$$

- *Soundness: For all non-uniform polynomial-time adversarial prover P^*, there exists a negligible function $\mu(\cdot)$ such that for every all $x \notin L$,*

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|x|}); (P^{*O}, V)(x, \mathsf{pp}) = 1] \leq \mu(|x|)$$

*Additionally, if the following condition holds, (P, V) is an **$\mathcal{O}$-oracle aided argument of knowledge**:*

- *Argument of knowledge: There exists a expected PPT algorithm E such that for every polynomial-size P^*, there exists a negligble function $\mu(\cdot)$ such that for every x,*

$$\begin{aligned}\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|x|}); w \leftarrow E^{P^{*O}(x,\mathsf{pp})}(x, \mathsf{pp}); w \in R_L(x)] \\ \geq \Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|x|}); (P^{*O}, V)(x, \mathsf{pp}) = 1] - \mu(|x|)\end{aligned}$$

Definition 11 (Oracle-aided Resettably-sound Interactive Arguments)**.** *An $\mathcal{O}$-oracle aided resetting attack of a cheating prover P^* on a resettable verifier V is defined by the following three-step random process, indexed by a security parameter n.*

1. *An initial setup where a public parameter and an oracle are generated: $\mathsf{pp}, O \leftarrow \mathcal{O}(1^n)$. P^* is given pp and oracle access to O.*
2. *Uniformly select and fix $t = poly(n)$ random-tapes, denoted $r_1, \ldots, r_t$, for V, resulting in deterministic strategies $V^{(j)}(x) = V_{x,r_j}$ defined by $V_{x,r_j}(\alpha) = V(x, r_j, \alpha)$, where $x \in \{0,1\}^n$ and $j \in [t]$. Each $V^{(j)}(x)$ is called an incarnation of V .*
3. *On input 1^n, machine P^* is allowed to initiate $poly(n)$-many interactions with the $V^{(j)}(x)$'s. The activity of P^* proceeds in rounds. In each round P^* chooses $x \in \{0,1\}^n$ and $j \in [t]$, thus defining $V^{(j)}(x)$, and conducts a complete session with it.*

Let (P, V) be an $\mathcal{O}$-oracle aided interactive argument for a language L. We say that (P, V) is an $\mathcal{O}$-oracle aided resettably-sound argument *for L if the following condition holds:*

- $\mathcal{O}$-oracle aided resettable soundness*: For every polynomial-size resetting attack, the probability that in some session the corresponding $V^{(j)}(x)$ has accepted and $x \notin L$ is negligible.*

Towards our goal of constructing of oracle-aided resettably-sound zero-knowledge, we now define and construct an oracle-aided version of universal arguments.

4.1 Oracle-aided Universal Arguments

Universal arguments (introduced in [BG02] and closely related to CS-proofs [Mic00]) are used in order to provide "efficient" proofs to statements of the form $y = (M, x, t)$, where y is considered to be a true statement if M is a non-deterministic machine that accepts x within t steps. The corresponding language and witness relation are denoted $L_{\mathcal{U}}$ and $\mathbf{R}_{\mathcal{U}}$ respectively, where the pair $((M, x, t), w)$ is in $\mathbf{R}_{\mathcal{U}}$ if M (viewed here as a two-input deterministic machine) accepts the pair (x, w) within t steps. Notice that every language in NP is linear time reducible to $L_{\mathcal{U}}$. Thus, a proof system for $L_{\mathcal{U}}$ allows us to handle all NP-statements. In fact, a proof system for $L_{\mathcal{U}}$ enables us to handle languages that are beyond NP, as the language $L_{\mathcal{U}}$ is NE-complete (hence the name universal arguments).[9] We here provide an oracle-aided variant of the [BG02] definition of universal arguments.

Definition 12 (Oracle-aided Universal Argument)**.** *An oracle-aided protocol (P, V) is called an $\mathcal{O}$-**oracle-aided universal argument** system if it satisfies the following properties:*

- Efficient verification: *There exists a polynomial p such that for any $y = (M, x, t)$, and for any pp, O generated by $\mathcal{O}$, the total time spent by the (probabilistic) verifier strategy V, on common input y, pp, is at most $p(|y| + |\mathsf{pp}|)$. In particular, all messages exchanged in the protocol have length smaller than $p(|y| + |\mathsf{pp}|)$.*

- Completeness with a relatively efficient oracle-aided prover: *For every $(y = (M, x, t), w)$ in $\mathbf{R}_{\mathcal{U}}$,*

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^{|y|}); (P^O(w), V)(y, \mathsf{pp}) = 1] = 1.$$

  Furthermore, there exists a polynomial q such that the total time spent by $P^O(w)$, on common input $y = (M, x, t)$, pp, is at most $q(T_M(x, w) + |\mathsf{pp}|) \leq q(t + |\mathsf{pp}|)$, where $T_M(x, w)$ denotes the running time of M on input (x, w).

- Weak proof of knowledge for adaptively chosen statements: *For every polynomial p there exists a polynomial p' and a probabilistic polynomial-time oracle machine E such that the following holds: for every non-uniform polynomial-time oracle algorithm P^*, if*

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^n); R \leftarrow \{0,1\}^\infty; y \leftarrow P_R^{*O}(\mathsf{pp}) : (P_R^{*O}(\mathsf{pp}), V(y, \mathsf{pp})) = 1] > 1/p(n)$$

  then

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^n); R, r \leftarrow \{0,1\}^\infty; y \leftarrow P_R^{*O}(\mathsf{pp}) : \exists w = w_1, \ldots w_t \in \mathbf{R}_{\mathcal{U}}(y) \text{ s.t. } \forall i \in [t], E_r^{P_R^{*O}}(\mathsf{pp}, y, i) = w_i] > \frac{1}{p'(n)}$$

  where $\mathbf{R}_{\mathcal{U}}(y) \stackrel{\text{def}}{=} \{w : (y, w) \in \mathbf{R}_{\mathcal{U}}\}$.

Note that our proof of knowledge condition is somewhat different from the one used in [BG02] in that we allow the (cheating) prover to *adaptively* choose the statement to be proved, after having seen the public parameter, and having interacted with its oracle.

Nevertheless, as we shall see, the construction of [BG02] and their analysis will be useful to us. Recall that in the construction of [BG02] tree hashing is used to hash down a "long" PCP proof into

[9] Furthermore, every language in NEXP is polynomial-time (but not linear-time) reducible to $L_{\mathcal{U}}$

a fixed-length "tree root"; the soundness property relies on collision resistant of this tree hashing. Let SIG' be a canonical sig-com scheme with $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ and Com being its underlying signature scheme and commitment scheme. We observe that if we replace the use of tree hashing in [BG02] scheme with a sig-com tree using SIG', then the resulting protocol is an $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument for the following signature oracle $\mathcal{O}^{\mathsf{SIG}}$.

Definition 13 (Signature Oracle)**.** *Given a signature scheme* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$*, we define a signature oracle* $\mathcal{O}^{\mathsf{SIG}}$ *as follows: On input a security parameter* n*,* $\mathcal{O}^{\mathsf{SIG}}(1^n)$ *generates* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$ *and lets* $\mathsf{pp} = \mathsf{vk}$ *and* $O(m) = \mathsf{Sign}_{\mathsf{sk}}(m)$ *for every* $m \in \{0,1\}^{\mathrm{poly}(n)}$*.*

In fact, the universal argument has an even stronger completeness property that will be useful for us: completeness hold even if the prover only gets access to a sig-com oracle (instead of a signature oracle), and even if this is an *arbitrary* (not necessarily using the honest sign and commit algorithms) sig-com oracle, as long as the oracle outputs valid sig-com's (for messages of a certain fixed length) with overwhelming probability. More formally,

Definition 14 (Valid Sig-com Oracle)**.** *An oracle* $\mathcal{O}'$ *is a* ***valid*** (SIG', ℓ) ***oracle*** *if there is a negligible* $\mu(\cdot)$ *such that for every* $n \in N$*, the following holds with probability* $1 - \mu(n)$ *over* $\mathsf{pp}, O \leftarrow \mathcal{O}'(1^n)$*: for every* $m \in \{0,1\}^{\ell(n)}$*,* $O(m)$ *returns* (σ, τ) *such that* $\mathsf{Ver}'_{\mathsf{vk}}(m, \sigma, \tau) = 1$ *with probability at least* $1 - \mu(n)$*.*

We note that oracles that use arbitrarily biased randomness for commitments are also considered *valid* sig-com oracles. (These are precisely the kind of oracles we will be forced to use later on).

Definition 15. *An* $\mathcal{O}^{\mathsf{SIG}}$*-aided universal argument* (P, V) *has* (SIG', ℓ)***-completeness*** *if there exists a prover* P' *such that the completeness condition holds for* (P', V) *when the oracle* $\mathcal{O}^{\mathsf{SIG}}$ *is replaced by any valid* (SIG', ℓ) *oracle* $\mathcal{O}'$*.*

We now have the following theorem.

Theorem 7. *Let* SIG' *be a canonical sig-com scheme with* SIG *and* Com *being its underlying signature scheme and commitment scheme. Then there exists a* (SIG', ℓ)*-complete* $\mathcal{O}^{\mathsf{SIG}}$*-aided universal argument with* $\ell(n) = 2n$*.*

The proof of the theorem identically follows that of Barak and Goldreich [BG02], with a minor modification to deal with adaptively chosen statements when proving the weak argument of knowledge property. For completeness, we provide a full proof in Appendix B (very closely following the presentation of [BG02]).

4.2 Oracle-aided Zero-Knowledge Protocols

We now turn to constructing oracle-aided resettably-sound zero-knowledge protocols. We start by defining a strong notion of an $\mathcal{O}$-oracle-aided version of ZK. First of all, we restrict to protocols where the honest prover does not access the oracle. Secondly, we require that simulation can be performed given oracle access to *any* valid SIG' oracle. These two restrictions will be important when we later instantiate the oracle-aided protocol in the plain model.

Definition 16 (Oracle-aided Zero-Knowledge)**.** *A pair of algorithms (P, V) is* (SIG', ℓ)**-oracle aided zero-knowledge** *for a NP language L with witness relation R_L if for every polynomial-time adversarial verifier V^*, there exists a simulator S, such that for every valid (SIG', ℓ)-oracle $\mathcal{O}'$, the following ensembles are indistinguishable over $x \in L$,*

$$\{\mathsf{pp}, O \leftarrow \mathcal{O}'(1^{|x|}) : (\mathsf{pp}, \mathsf{View}_{V^*}(P(w), V^*(z))(x, \mathsf{pp}))\}_{x \in L, w \in R_L(x), z \in \{0,1\}^*}$$
$$\approx \{\mathsf{pp}, O \leftarrow \mathcal{O}'(1^{|x|}) : (\mathsf{pp}, S^O(x, z, \mathsf{pp}))\}_{x \in L, w \in R_L(x), z \in \{0,1\}^*}$$

We now turn to the question of constructing a protocol that satisfies the above requirements. Note that, as a first attempt, we could try constructing a constant-round public-coin ZK protocol by replacing the tree hashing in Barak's protocol [Bar01] with sig-com trees, and then apply the PRF transformation of [BGGL01] to achieve resettable soundness. While this indeed could be used to get a resettably-sound ZK protocol in the oracle-aided model, the resulting protocol would require the honest prover to make polynomially many queries to the oracle (to complete the WIUARG). To get around this, we instead rely on a variant of Barak's protocol used in Pass and Rosen [PR05], which provides a "special-purpose" implementation of the WIUARG used in Barak's protocol in which the honest prover does not need to perform any "hashing".[10]

More precisely, our protocol proceeds as follows. In Stage 1, P sends a commitment $c = \mathsf{Com}(0^{2n})$, and then V sends back a challenge $r \in \{0,1\}^n$ as in Barak's protocol. In Stage 2, P and V first execute an "encrypted" universal argument $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ of the statement that *"c is a commitment to a sig-com tree root of a program M and $M(c) = r$,"* where instead of sending the message in the clear, the prover sends commitments to the messages. The honest prover simply sends commitments to 0 (and thus will fail in this encrypted universal argument). Finally, P and V execute a witness-indistinguishable argument of knowledge of the statement that *"$x \in L$ OR V_{UA} accepts in the encrypted universal argument.*

A formal description of the protocol can be found in Fig. 1 and Fig. 2. Note that, in this construction, the honest prover P can convince the verifier by proving $x \in L$ in the final witness indistinguishable argument without making any oracle queries. This leads to the following theorem. The proof of the theorem closely follows [Bar01, PR05] but the proof of the "argument of knowledge" property requires special care to deal with the fact that a cheating prover may adaptively choose the statements to be proved in the encrypted universal argument (after having interacted with its oracle).[11] A formal proof of the theorem can be found in Appendix C.

Theorem 8. *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists an $\mathcal{O}^{\mathsf{SIG}}$-oracle aided argument of knowledge (P, V) for NP; additionally,*

1. *(P, V) is constant-round and public-coin;*
2. *P does not make any queries to its oracle;*
3. *(P, V) is (SIG', ℓ)-oracle-aided zero-knowledge for $\ell(n) = 2n$.*

[10] In fact, early versions of Barak's protocol also relied on such a special-purpose implementation of WIUARG.

[11] In [Bar01, PR05] these issue does not arise since different, independently chosen hash-functions are used in Stage 1 and in Stage 2.

Common Input: An instance x of a language $L \in \mathsf{NP}$ with witness relation $\mathbf{R}_L$.

Auxiliary input to P: A witness w such that $(x, w) \in \mathbf{R}_L$.

Primitives Used: A canonical sig-com scheme SIG' with SIG and Com as the underlying signature and commitment schemes, and a (SIG', ℓ)-complete $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ with $\ell(n) = 2n$.

Set Up: Run $(\mathsf{pp}, O) \leftarrow \mathcal{O}^{\mathsf{SIG}}(1^n)$, add pp to common input for P and V. Furthermore, allow P oracle access to O.

Stage One (Trapdoor):

P_1: Send $c_0 = \mathsf{Com}(0^{2n}, \tau_0)$ to V with uniform τ_0

V_1: Send $r \xleftarrow{\$} \{0,1\}^n$ to P

Stage Two (Encrypted Universal Argument):

P_2: Send $c_1 = \mathsf{Com}(0^{2n}, \tau_1)$ for uniformly selected τ_1

V_3: Send r', uniformly chosen random tape for V_{UA}

P_3: Send $c_2 = \mathsf{Com}(0^k, \tau_2)$ for uniformly selected τ_2, where k is the length of P_{UA}'s second message.

Stage Three: (Main Proof)

$P \Leftrightarrow V$: A WI-AOK $\langle P_{\mathsf{WI}}, V_{\mathsf{WI}} \rangle$ proving the OR of the following statements:

1. $\exists\ w \in \{0,1\}^{\mathrm{poly}(|x|)}$ s.t. $(x, w) \in \mathbf{R}_L$.
2. $\exists\ \langle p_1, p_2, \tau_1, \tau_2 \rangle$ s.t. $(\langle c_0, r, c_1, c_2, r', \mathsf{pp} \rangle, \langle p_1, p_2, \tau_1, \tau_2 \rangle) \in \mathbf{R}_{L_2}$ (defined in Fig. 2).

Figure 1: $\mathcal{O}^{\mathsf{SIG}}$-aided ZK Argument of Knowledge.

Relation 1: Let SIG' a sig-com scheme, with underlying signature scheme SIG and commitment scheme Com. Let $\mathtt{ECC}$ be a binary error-correcting code with constant min-distance and efficient encoding algorithm. We say that $\langle c_0, r, \mathsf{pp}\rangle \in L_1$ if $\exists\langle \tau_0, d, l_\lambda, C, \{\vec{\rho_i}\}_{i\in[2^d]}\rangle$ such that

- $c_0 = \mathsf{Com}((d, l_\lambda), \tau_0)$
- (d, l_λ) are the depth and root of a sig-com tree for C w.r.t. pp
- Each $\vec{\rho_i}$ is a valid sig-com path for leaf i of this sig-com tree. That is, $\mathsf{PATH}^{\mathsf{SIG}'}(\vec{\rho_i}, C_i, i, l_\lambda, \mathsf{pp}) = 1$ for each i.
- $C = \mathtt{ECC}(\Pi)$ for some circuit Π
- $\Pi(c_0) = r$.

We let $\mathbf{R}_{L_1}$ be the witness relation corresponding to L_1.

Relation 2: Let L_1 be described as above, with respect to SIG' and $\mathtt{ECC}$. Let $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ be a (SIG', ℓ)-complete $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument with $\ell(n) = 2n$. We say that $\langle c_0, r, c_1, c_2, r', \mathsf{pp}\rangle \in L_2$ if $\exists\langle p_1, p_2, \tau_1, \tau_2\rangle$ such that

- $c_1 = \mathsf{Com}(p_1, \tau_1)$, $c_2 = \mathsf{Com}(p_2, \tau_2)$.
- (p_1, r', p_2) constitutes an accepting $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ transcript for $\langle c_0, r\rangle \in L_1$.

We let $\mathbf{R}_{L_2}$ be the witness relation corresponding to L_2.

Figure 2: Relations used in the $\mathcal{O}^{\mathsf{SIG}}$-aided ZK protocol in Fig. 1.

Finally, we apply the PRF transformation of [BGGL01] to the $\mathcal{O}^{\mathsf{SIG}}$-oracle aided ZK protocol (P, V) constructed above to achieve $\mathcal{O}^{\mathsf{SIG}}$-oracle aided resettable soundness More precisely, we modify the public-coin verifier V to a "PRF-verifier" $\tilde{V}$ that samples a seed s for a PRF f_s at beginning and then generates each verifier message by applying f_s to the current transcript. The proof in [BGGL01] relativizes and as a consequence we have the following theorem:

Theorem 9. *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists an $\mathcal{O}^{\mathsf{SIG}}$-aided constant-round resettably-sound argument of knowledge (P, V) for NP; additionally,*

1. *P does not make any queries to its oracle;*
2. *(P, V) is (SIG', ℓ)-oracle-aided zero-knowledge for $\ell(n) = 2n$.*

5 Resettably-sound Zero Knowledge in the Plain Model

Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Let (P, V) be an $\mathcal{O}^{\mathsf{SIG}}$-aided resettably sound argument of knowledge for a language L with witness relation R_L, where P does not make any queries to its oracle. Consider the protocol $(\tilde{P}, \tilde{V})$ that on common input x, and auxiliary prover input w proceeds as follows.

1. Init: $\tilde{V}$ runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^n)$ and sends vk to $\tilde{P}$.
2. Signing Slot:

- $\tilde{P}$ generates $c = \mathsf{Com}(0^{2n}; \tau)$, where τ is uniformly sampled, and sends c to $\tilde{V}$.
- $\tilde{V}$ replies with $\sigma = \mathsf{Sign}_{\mathsf{sk}}(c)$.
- $\tilde{P}$ aborts if σ is not a valid signature of c.

3. Body: Invoke the protocol $(P(w), V)(x, \mathsf{pp})$ with $\mathsf{pp} = \mathsf{vk}$.

Lemma 10. *If (P, V) is $(\mathsf{SIG}', 2n)$-oracle-aided zero-knowledge for L with witness relation R_L, then $(\tilde{P}, \tilde{V})$ is a fixed-input resettably-sound zero-knowledge argument of knowledge for L with witness relation R_L.*

Note that here we only obtain a *fixed-input* resettably sound argument of knowledge (defined in Definition 3), but this can be transformed into a "full-fledged" resettably sound one by using the transformation in Claim 2, which thus establishes our main Theorem 1.

Before proving Lemma 10 formally, we provide a high-level sketch first. Completeness of $(\tilde{P}, \tilde{V})$ follows directly from the completeness of (P, V). Resettable-soundness and the argument of knowledge property, roughly speaking, follow by emulating all signature slot messages using the oracle. The zero-knowledge simulator proceeds by first honestly emulating the signature slot for the malicious verifier V^*, and if V^* provides an accepting signature, we next run the oracle-aided simulator, and appropriately rewinding the malicious verifier during the signature slot to appropriately implement *some* valid sig-com oracle. The verifier may not always answer, but we can "keep rewinding" him, sending fresh commitments until he does. Roughly speaking, the key point is that if V^* did provide a valid signature during the first pass, then in expectation, by the hiding property of the commitment scheme, we only need a polynomial number of rewindings. This "almost" works: just as in [GK96], we need to take special care to deal with verifier's that only provide valid signatures with very small probability. We proceed with a formal proof.

Proof. Completeness of $(\tilde{P}, \tilde{V})$ follows directly from the completeness of (P, V) since by assumption, P never makes any oracle queries.

To prove the fixed-input resettable-soundness of $(\tilde{P}, \tilde{V})$, we show how to convert a malicious prover $\tilde{P}^*$ for $(\tilde{P}, \tilde{V})$ into an *oracle-aided* malicious prover P^* for (P, V) that succeeds with the same probability. $P^{*O}(1^n, \mathsf{pp})$ internally emulates an execution of $\tilde{P}^*$ as follows:

- Upon invocation P^* feeds $\tilde{P}^*$ the message pp (corresponding to the "Init message" of the protocol.
- Whenever $\tilde{P}^*$ makes a signing slot query, that is, whenever it requests a signature on some message c from $\tilde{V}$, P^* forwards c to its oracle O, and relays the answer back to $\tilde{P}^*$ as $\tilde{V}$'s reply.
- All other messages are forwarded externally to the verifier, and the verifier's replies are relayed back.

It follows by inspection that P^* succeeds in convincing V (during a reset attack) with identically the same probability as $\tilde{P}^*$ convinces $\tilde{V}$, since the view of $\tilde{P}^*$ in the emulation by P^* is identical to its view in the execution with $\tilde{V}$.

By the same argument we have that $(\tilde{P}, \tilde{V})$ is an argument of knowledge: Let E be the extractor for (P, V), and define the extractor $\tilde{E}$ for $(\tilde{P}, \tilde{V})$ that given oracle access to $\tilde{P}^*$, proceeds as follows: $\tilde{E}$ runs $\mathsf{pp}, O \leftarrow \mathcal{O}$, and then $\tilde{E}$ emulates the execution of E given oracle access to P^* described

above, while 1) internally emulating all oracle queries by P^* (using O) and 2) externally querying (and relaying back the answer) $\tilde{P}^*$ on all queries made by E to P^*. Since $\tilde{P}^*$ succeeds in convincing $\tilde{V}$ with identically the same probability as P^* convinces V, it follows by the argument of knowledge property of (P, V) that $(\tilde{P}, \tilde{V})$ also is an argument of knowledge.

Let us turn to zero-knowledge. Consider some malicious (w.l.o.g. deterministic) verifier $\tilde{V}^*$ for $(\tilde{P}, \tilde{V})$ of size $T_{\tilde{V}^*}$. We construct a simulator $\tilde{S}$ for $\tilde{V}^*$. Roughly speaking, $\tilde{S}$ starts by simulating $(\tilde{P}, \tilde{V}^*)$ honestly up to the end of the Signing Slot, and if $\tilde{P}$ does not abort, $\tilde{S}$ continue to simulate the view of $\tilde{V}^*$ in the Body part by 1) viewing the "residual" $\tilde{V}^*$ as a malicious V^* for (P, V), 2) preparing a valid $(\mathsf{SIG}', 2n)$ oracle $\mathcal{O}'$ (by rewinding $\tilde{V}^*$ at the Signing Slot in the spirit of Goldreich-Kahan [GK96]), and 3) invoking the simulator S for V^* with oracle $\mathcal{O}'$.

More precisely, $\tilde{S}$ first receives vk from $\tilde{V}^*$, generates and sends to $\tilde{V}^*$ an honest commitment $c = \mathsf{Com}(0^{2n}; \tau)$ with uniform τ, and then receives back a signature σ from $\tilde{V}^*$. If σ is not a valid signature of c, then the simulation halts immediately and outputs the transcript upto that point. Otherwise, let V^* be the residual $\tilde{V}^*$ at the end of the Signing Slot (which is a malicious verifier for (P, V)), and construct an oracle $\mathcal{O}'$ as follows.

- $\tilde{S}$ repetitively queries $\tilde{V}^*$ at the Signing Slot with fresh commitments $\mathsf{Com}(0^{2n}; \tau)$ until it collects $2n$ valid signatures. Let t be the number of queries $\tilde{S}$ makes.
- Define $\mathcal{O}'$ that outputs $\mathsf{pp} = \mathsf{vk}$, and an oracle O that on input a message $m \in \{0,1\}^{2n}$, proceeds as follows: O repetitively queries $\tilde{V}^*$ at the Signing Slot with fresh commitments $\mathsf{Com}(m; \tau)$ for at most t times. If $\tilde{V}^*$ ever replies a valid signature σ for $\mathsf{Com}(m, \tau)$, then O outputs (σ, τ). Otherwise, O returns $\perp$.

If $t \geq 2^{n/2}$, then $\tilde{S}$ aborts. Otherwise, $\tilde{S}$ invokes the simulator S for V^* with oracle $\mathcal{O}'$, while emulating the oracle for S during its execution, and outputs the view of V^* (which is also a view of $\tilde{V}^*$) generated by S at the end.

To analyze $\tilde{S}$, we introduce some notation. Let $p(m)$ be the probability that $\tilde{V}^*$ on query a random commitment $c = \mathsf{Com}(m, \tau)$ of $m \in \{0,1\}^{2n}$ at the Signing Slot, returns a valid signature of c. Let $p = p(0^{2n})$.

We first show that $\tilde{S}$ runs in expected polynomial time. To start, note that $\tilde{S}$ aborts at the end of the Signature Slot with probability $1 - p$, and in this case, $\tilde{S}$ runs in polynomial time. With probability p, $\tilde{S}$ continues to invoke a strictly polynomial-time simulator S for the residual V^*, which has size bounded by $T_{\tilde{V}^*}$. Thus, S runs in some $T = \mathrm{poly}(T_{\tilde{V}^*})$ time and makes at most T queries to its oracle O, which in turn runs in time $t \cdot \mathrm{poly}(n)$ to answer each query. Also note that $\tilde{S}$ runs in time at most 2^n, since $\tilde{S}$ aborts when $t \geq 2^{n/2}$. Now, we claim that $t \leq 10n/p$ with probability at least $1 - 2^{-n}$, and thus the expected running time of $\tilde{S}$ is at most

$$(1-p) \cdot \mathrm{poly}(n) + p \cdot T \cdot (10n/p) \cdot \mathrm{poly}(n) + 2^{-n} \cdot 2^n \leq \mathrm{poly}(T_{\tilde{V}^*}, n).$$

To see that $t \leq 10n/p$ with overwhelming probability, let $X_1, \ldots, X_{10n/p}$ be i.i.d. indicator variables on the event that $\tilde{V}^*$ returns a valid signature for a random $\mathsf{Com}(0^{2n}; \tau)$, and note that $t \leq 10n/p$ implies $\sum_i X_i \leq 2n$, which by a standard Chernoff bound, can only happen with probability at most 2^{-n}.

Finally, we argue indistinguishability. First, the computational hiding property of Com implies that there exists some negligible $\nu(\cdot)$ such that $|p(m) - p| \leq \nu(n)$ for every $m \in \{0,1\}^{2n}$. Now we consider two cases. If $p \leq 2\nu$, then the indistinguishability trivially holds since the interaction aborts

at the end of the Signature Slot (in this case, the view is perfectly simulated) with all but negligible probability. On the other hand, if $p \geq 2\nu$, we show that $\mathcal{O}'$ generated by $\tilde{S}$ is a valid $(\mathsf{SIG}', 2n)$ oracle for SIG' with overwhelming probability, and thus the indistinguishability of $\tilde{S}$ follows by the indistinguishability of S.

To see that $\mathcal{O}'$ is a valid $(\mathsf{SIG}', 2n)$ oracle for SIG' with overwhelming probability, note again by a Chernoff bound that $n/p \leq t \leq 2^{n/2}$ with probability at least $1 - 2^{-\Omega(n)}$. In this case, for every $m \in \{0,1\}^{2n}$, $p(m) \geq p - \nu \geq p/2$ implies that $t \geq n/2p(m)$, and thus $O(m)$ learns a valid signature of $\mathsf{Com}(m;\tau)$ from $\tilde{V}^*$ with probability at least $1 - 2^{-\Omega(n)}$. □

6 Applications

6.1 Resettable Zero-knowledge and Resettable Witness-indistinguishability

By plugging our resettably-sound zero-knowledge argument of knowledge into the constructions of [CGGM00, PRS02, BGGL01], with some minor modifications that we discuss shortly, we immediately obtain the following theorem. (Roughly speaking, in a resettably-witness indistinguishable (resp., zero-knowledge) argument, the witness-indistinguishability (resp., zero-knowledge) property is required to hold also in the presence of a resetting verifier; see [CGGM00, BGGL01] for formal definitions.)

Theorem 11. *Assume the existence of one-way functions. Then*

- *there exists a constant-round resettably-witness-indistinguishable argument of knowledge for all of NP,*
- *there exists a $\tilde{O}(\log n)$-round resettable-zero-knowledge argument of knowledge for all of NP.*

The construction of a constant-round resettably-witness-indistinguishable argument of knowledge follows directly from the results of [BGGL01]. For the construction of a resettably-zero-knowledge argument of knowledge, recall that [BGGL01] (following [CGGM00]) construct resettably-zero-knowledge arguments of knowledge by compiling (using a resettably-sound zero-knowledge argument of knowledge) some underlying concurrent zero-knowledge protocols of the "committed-verifier type" where the verifier commits to its "challenges" at the beginning of the protocol, and then reveals them one by one in sequential "slots". The underlying concurrent zero-knowledge protocol, however, relies on commitment in use being statistically-hiding. We note that by a minor tweak of the concurrent zero-knowledge protocol of [PRS02], we can use also computationally-hiding commitments (that exists based on one-way functions). More precisely, we can replace the statistically-hiding commitment that the verifier uses with computationally-hiding commitments if the verifier in the final stage of the protocol, instead of opening up all commitments, simply reveals the committed values and provides a *resettably-sound* zero-knowledge argument of knowledge of the value. In fact, the same trick of replacing statistically-hiding commitments with computationally-hiding commitmetns, but using just a "plain", as opposed to resettably-sound, zero-knowledge argument of knowledge in the final stage in the protocol, was used in [PRS02] to get a concurrent zero-knowledge argument from one-way functions, and soundness follows in exactly the same way as in [PRS02] (but to preserve resettable-zero-knowledge, we here need to use a resettably-sound zero-knowledge argument of knowledge).

6.2 Simultaneously Resettability

We proceed to discuss how to remove the need for CRHs in the simultaneously resettable zero-knowledge argument protocol of Deng, Goyal and Sahai [DGS09] (referred to as DGS hereafter). Recall that simultaneous resettability here means that security (both zero-knowledge and soundness) holds even with respect to resetting attackers; see [BGGL01, DGS09] for a formal definition.

Theorem 12. *Assume the existence of one-to-one one-way functions and enhanced trapdoor permutations. Then there exists a simultaneously-resettable zero-knowledge argument for NP.*

We start by reviewing the construction of DGS, which proceeds in two steps. First, DGS constructs a "main protocol" that satisfies a weak notion of resettable soundness (called "hybrid soundness") and a weak notion of resettable zero-knowledge (called "relaxed concurrent zero-knowledge"). Then, general transformations are applied to compile the main protocol to a simultaneously resettable one. The second step relies only on the existence of a ZAPs [DN07] (which can be based on the existence of enhanced trapdoor permutations), whereas the first step additionally requires the existence of CRHs and a non-intertacive commitment scheme with a "unique decommitments" (which can be based on the existence of one-to-one one-way functions); see [DGS09] for more details. Therefore, to achieve our goal, it suffices to remove the use of CRHs (by replacing them with signature trees) in the construction of their main protocol.

In more detail, the main protocol of DGS has the following high-level structure.

1. The protocol start by having the prover P committing to $2n^2$ "challenges" $ch_1, \ldots, ch_{2n^2}$.
2. Then, the verifier V sends P a "trapdoor" $trap = \mathsf{Com}(1; \tau)$.
3. (P, V) next proceed with $2n^2$ "rewinding slots" as follows: for each $i \in [2n^2]$,
   - P sends ch_i to V.
   - (P, V) engage in a "special-purpose" resettably-sound zero-knowledge argument. where P proves to V that either ch_i is the correct challenge committed in the first prover message, or $x \in L$.
   - V replies an answer ans_i to ch_i if the rs-ZK argument is accepting.
4. Finally, P proves to V using a simultaneously resettable WI proof (which can be constructed using ZAPs) that either $x \in L$ or $trap = \mathsf{Com}(1; \tau)$ for some τ.

The $2n^2$ slots are used by the simulator S to extract a "fake witness" τ to complete the final WI (i.e., to "solve" the session), where S appropriately rewinds some of the slots *once* to collect two challenge-answer pairs $(ch_i, ans_i), (ch'_i, ans'_i)$ to extract τ. For the "honest" challenge ch_i (i.e., the one committed in the first prover message), S simply completes the rs-ZK honestly (i.e., using the honest prover strategy), whereas for "fake" challenges ch'_i, S relies on the rs-ZK simulator to convince the verifier to provide an answer.

In the above protocol, CRHs are used in the rs-ZK arguments (as in Barak's protocol) in each slot. We remove the use of CRHs by instantiating the rs-ZK argument with sig-com trees, in exactly the same way as Barak's protocol: More precisely, at the beginning of the protocol, the verifier is asked to pick a verification key vk, and then each rs-ZK protocol is instantiated using sig-com trees with respect to this verification key vk (and as before, we add a signature slot to the begining of

the rs-ZK protocol); we emphasize that the *same* verification key is used in all the different slots in the DGS protocol—this will faciliate the analysis.

Recall, however, that once we are using sig-com trees, our simulator needs to rewind the verifier polynomially many times to construct the sig-com trees (say, $g(n) = \text{poly}(|V^*|)$ signatures are needed, where $|V^*|$ is the size of the verifier[12]). Thus, we modify the DGS simulator in a straightforward way: rather than just rewinding each slot once, we additionally rewinding it $3g(n)$ more times to get the appropriate sig-com tree and finally simulate the rs-ZK protocol[13] (note that for this step to work, it is crucial that we use a single verification key vk in all the slots; otherwise, to get the appropriate sig-com tree we would have to make sure to rewind "just" the signature slot and not the "whole" DGS slot).

To see why this works, let us first briefly review the analysis in the original DGS protocol and then slightly generalize it to our context. Roughly speaking, DGS uses a variant of the Richardson-Killian (RK) [RK99] recursive simulation strategy, where for each session i, the simulator selects $2n$ slots to be rewound exactly once, but "cutting-off" the rewinding if the total number of new slots that open within the slot being rewound exceeds a certain threshold that depends on the number of nested rewindings (referred to as the "recursive level of the simulation"). The crux of the DGS analysis is showing that for the simulator to get "stuck" (i.e., not solving the session), there must be at least n out of the $2n$ slots where some "bad" event happens: namely, a) that the slot was "light" in the main thread (i.e., the number of new slots within it was below the threshold of the next recursive level), but b) the rewinding failed (i.e., it the slot became "heavy" during the rewinding). Additionally, for each such slot, *independently*, this bad event happens with probability at most $1/4 + \mathsf{ngl}(n)$ (this follows since the messages sent by the simulator in the main thread and the rewinding are computationally indistinguishable); thus, the chance of getting stuck on each session is at most 2^{-n}.

In our context, the definition of a bad event is slightly different: the bad event happens for a slot if a) the slot was "light" on the main-thread, and *either* b) the slot became "heavy" during the first rewinding *or* c) the slot was only "light" in $< g(n)$ out of $3g(n)$ additional rewindings[14] (in preparing the sig-com tree) of the $3g(n)$ rewindings. It follows by identically the same argument as in DGS, that our simulator only get stuck on a session i if there exists at least n slots (out of the $2n$ slots being rewound) for which this bad event happens.

Below, we argue that for each slot being rewound, the bad event happens with probability at most $1/2$; as before, this suffices to conclude that the simulation gets stuck on the current session with probability 2^{-n}. Let p be the probability of the slot being "light" in the main thread (which implies that the slots is "light" in the rewindings with probability $p \pm \mathsf{ngl}(n)$). By definition, the bad event thus happens with probability at most p. Thus, if $p < 1/2$, then we are done. On the other hand, if $p \geq 1/2$, then condition b) happens with probability at most $1 - p + \mathsf{ngl}(n)$, and by a Chernoff bound, c) happens with negligible probability. Thus, it follows by a union bound that the bad event happens with probability at most $p \cdot (1 - p - \mathsf{ngl}(n)) + p \cdot \mathsf{ngl}(n) \leq 1/2$.

To summarize, by instantiating the rs-ZK arguments with sig-com trees we remove the use of

[12] The size of the trees are proportional to the size of the committed program M and the complexity of the universal argument respectively, both of which as demonstrated by DGS, are both upper bounded by $\text{poly}(|V^*|)$.

[13] Technically, the first rewinding gets "paused" when we have to commit to the program M, and once we have to perform the universal argument in order for the simulator to generate sig-com tree, which is does through all the other rewindings.

[14] Recall that we need $g(n)$ rewindings to prepare the sig-com tree, and one additional rewinding—the first one—to get back the answer from the verifier once we can simulate the rs-ZK argument.

CRHs in the main protocol of [DGS09]. By further plugging in the general transformations in the second step of [DGS09], we obtain simultaneously resettable zero-knowledge arguments for NP based on the existence of one-way permutations and trapdoor permutations.

6.3 Simultaneously Resettable Zero-Knowledge for NP from One-way Functions (Results From January 15th, 2013)

In this section, we show how to appropriately modify the above protocol to get an instantiation based only on one-way functions. These added results were established subsequently to the results of [OV12] and [CP13] and we here rely on these results. Let us also point out that, independently [BP12b], have also shown how to use the result from [CP13] within their framework to establish the same result as we show here.

Theorem 13. *Assume the existence of one-way functions. Then there exists a simultaneously-resettable zero-knowledge argument for NP.*

As mentioned, the need of trapdoor permutations in the DGS protocol is required for implementing ZAPs. As pointed out in the very recent work of Ostrovsky and Visconti, the only property needed from these ZAPs is that they are simultaneously-resettable witness indistinguishable (sr-WI) proofs,[15] and as such, any instantiation of a sr-WI proof would work in the protocol of DGS. Additionally, [OV12] show that it in fact suffices to replace these ZAPs also with a sr-WI arguments of knowledge. The work of Ostrovsky and Visconti also provide a construction of a sr-WI argument of knowledge based on the existence of collision-resistant hash functions, and use this construction to get a sr-ZK assuming the existence of collision-resistant hash functions and one-to-one one-way functions;[16] even more recently, Chung and Pass provide a construction of sr-WI arguments of knowledge based on one-way functions [CP13]. Thus, if we replace the use of ZAPs in the protocol described in Section 6.3 with the sr-WI argument of knowledge construction due to [CP13], we directly have a sr-ZK based on the existence of one-to-one one-way functions.

Below, we discuss how the need for one-to-one one-way functions can be replaced with just (plain) one-way functions. In fact, the need for one-to-one one-way functions is only implicit in the work DGS (but is explicitly pointed out in the presented construction there). As mentioned above, the need for one-to-one one-way function stems from the use of a non-interactive commitment scheme with a unique decommitment. Indeed, in personal communication, the authors of DGS have notified us that although the published version of their paper indeed relies on one-to-one one-way functions, they also had an alternative construction which doesn't need this additional assumption.

For completeness, we here provide our own modifications of the DGS protocol that dispenses of the need for one-to-one one-way functions (and thus yields the desired construction of a sr-ZK from one-way functions).

Recall that DGS relies on a mixture of a black-box and a non-black-box simulation strategy: The simulator S performs a variant of a "RK-style" [RK99] rewinding strategy to resolve each session, (i.e., extract a "fake" witness τ to a trapdoor $trap = \mathsf{Com}(1;\tau)$), and when a slot i is rewound, S sends a random "fake" challenge ch_i' and relies on the non-black-box rs-ZK simulator to convince the verifier to provide an answer ans_i' (whereas in the main thread, S sends the "honest" challenge ch_i and honestly completes the rs-ZK). At a high level, an issue with the non-black-box

[15] The fact that ZAPs are sr-WI was already noted in [BGGL01].

[16] As in [DGS09], the need for one-to-one one-way function is only implicit in [OV12].

simulation strategy is that the running-time of the non-black-box simulator grows exponentially with the number of "nested" sessions. Intuitively, this should not pose a serious issue since the RK-style rewinding pattern ensures the nesting depth is constant. But formalizing this intution becomes somewhat tricky. In particular, (the published version of) DGS required the use of non-interactive commitment scheme with a unique decommitment to formalize it (their formalization, requires to allow the simulator to commit to an "oracle" algorithm M, that gets access to a "decommitment oracle" O).

We here provide an alternative approach. In our approach, we do not require a decommitment oracle. Rather, when performing the non-black-box simulation, we ask the simulator to not only commit to the code of the verifier, but also to the code of the simulator;[17] see [CLP12] for a formalization of this.[18] As mentioned above, intuitively, this should lead to a polynomial-time simulation strategy since by the RK-style rewinding pattern, we only have a constant number of recursive rewinding, and thus only a constant number of nested non-black-box simulations. But there is an issue with this idea. The problem is that although there is only a constant number of nested recursive rewindings, messages from a higher-level (in the recursion) non-black-box simulation could appear in a lower-level simulation, leading to a circularity which in turn yields a blow-up in the running-time. (More precisely, we can not guarantee that the i-level recursive call takes time that is polynomial in the $(i+1)$-level recursive call, since the $(i+1)$-level recursive call may need to simulate messages from the i'th level). This issue can be easily resolved by noting that since there are only a constant number of recursive level, and each instance of the rs-ZK protocol only has a constant number of messages, such "higher-level" messages can only "ruin" the RK-style simulation for a constant number of slots. More precisely, whereas before DGS "cut-off" a rewinding of a slot if the slot was "too heavy" (i.e., number of slots inside it becomes too "large", where "large" is defined based on the recursive level), we here also cut-off the rewinding if a message from a higher-level non-black-box simulation needs to be sent. With this modification, we can ensure that the running-time is polynomial (there are a constant number of recursive calls, and each i-level recursive call takes time that is polynomial in the $(i+1)$-level recursive call). It just remains to argue that the simulation still succeeds. DGS argues that for each session, there exists some recursive level with at least n "light" slots, which suffices to conclude that the simulation succeeds; now, we only have $n - O(1)$ "good" slots (i.e., slots that are both short and do not contain any higher-level non-black-box simulation messages), but this still suffices to show that the simulation succeeds with high probability (in fact, having $\omega(\log n)$ such slots would suffice).

Acknowledgements

We are very grateful to Ran Canetti for pointing out the connection to [CGH04].

[17] DGS in fact also asks the simulator to commit to its own code, but only in a very partial way; more specifically, they only ask the simulator to commit to the part of its whose running time is a-priori bounded by a polynomial, and use the "decommitment oracle" to deal with the rest of the simulation.

[18] As discussed in [CLP12] and also [CLP13] there are some "randomness dependency" issues when doing this. In particular, if we commit to the code of the simulator, we are also committing to the randomness used to generate the commitment, and then it is no longer clear that indistinguishability of the commitment holds. Nevertheless, this circularity issue can be resolved in a rather straight-forward way as in [CLP12, CLP13].

References

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS '01*, pages 106–115, 2001.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Computational Complexity*, pages 162–171, 2002.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS'02*, pages 116–125, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

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

[BP12b] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. Cryptology ePrint Archive, Report 2012/729, 2012.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC '00*, pages 235–244, 2000.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *TCC*, pages 40–57, 2004.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC '01*, pages 570–579, 2001.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Advances in CryptologyâĂŤEUROCRYPT 2001*, pages 93–118, 2001.

[CLP12] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from falsifiable assumptions. Cryptology ePrint Archive, Report 2012/563, 2012. http://eprint.iacr.org/.

[CLP13] Ran Canetti, Huijia Lin, and Omer Paneth. Public coin concurrent zero-knowledge in the global hash model. To appear in TCC 2013, 2013.

[CP13] Kai-Min Chung and Rafael Pass. Simultaneous resettable wi from one-way functions. Cryptology ePrint Archive, Report 2013/010, 2013. http://eprint.iacr.org/.

[DGS09] Y. Deng, V. Goyal, and A. Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 251–260. IEEE, 2009.

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

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90*, pages 416–426, 1990.

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

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[LP11] Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.

[Mer89] R.C. Merkle. Digital signature system and method based on a conventional encryption function, November 14 1989. US Patent 4,881,264.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

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

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC '89*, pages 33–43, 1989.

[OV12] Rafail Ostrovsky and Ivan Visconti. Simultaneous resettability from collision resistance. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:164, 2012.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Theory and Computing Systems, 1993*, pages 3–17, 1993.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC '04*, pages 232–241, 2004.

[PR05] Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC '05*, pages 533–542, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS '02*, pages 366–375, 2002.

[PTV08] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishan Venkitasubramaniam. Concurrent zero knowledge: Simplifications and generalizations. Manuscript, 2008. http://hdl.handle.net/1813/10772.

[PTW09] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. In *CRYPTO '09*, pages 160–176, 2009.

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

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures, 1990.

A Additional Preliminaries

A.1 Computational Indistinguishability

The following definition of computational indistinguishability originates in the seminal paper of Goldwasser and Micali [GM84]. Let X be a countable set of strings. A **probability ensemble indexed by** X is a sequence of random variables indexed by X. Namely, any element of $A = \{A_x\}_{x\in X}$ is a random variable indexed by X.

Definition 17 (Indistinguishability)**.** *Let X and Y be countable sets. Two ensembles $\{A_{x,y}\}_{x\in X,y\in Y}$ and $\{B_{x,y}\}_{x\in X,y\in Y}$ are said to be **computationally indistinguishable over** X, if for every probabilistic machine D (the distinguisher) whose running time is polynomial in its first input, there exists a negligible function $\nu(\cdot)$ so that for every $x \in X, y \in Y$:*

$$|\Pr[D(x, y, A_{x,y}) = 1] - \Pr[D(x, y, B_{x,y}) = 1]| < \nu(|x|)$$

(In the above expression, D is simply given a sample from $A_{x,y}$ and $B_{x,y}$, respectively.)

A.2 Interactive Arguments

Definition 18 (Interactive Arguments)**.** *A pair of interactive algorithms (P, V) is an **interactive argument** for a NP language L with witness relation R_L if it satisfies the following properties:*

- *Completeness: There exists a negligible function $\mu(\cdot)$, such that for all $x \in L$, if $w \in R_L(x)$,*

$$\Pr[(P(w), V)(x) = 1] = 1 - \mu(|x|)$$

- *Soundness: For all non-uniform polynomial-time adversarial prover P^*, there exists a negligible function $\mu(\cdot)$ such that for every all $x \notin L$,*

$$\Pr[(P, V)(x) = 1] \leq \mu(|x|)$$

*If the following condition holds, (P, V) is an **argument of knowledge**:*

- *Argument of knowledge: There exists an expected PPT algorithm E such that for every polynomial-size P^*, there exists a negligble function $\mu(\cdot)$ such that for every x,*

$$\Pr[w \leftarrow E^{P^*(x)}(x); w \in R_L(x)] \geq \Pr[(P^*, V)(x) = 1] - \mu(|x|)$$

A.3 Witness Indistinguishability

An interactive protocol is **witness indistinguishable** (WI) [FS90] if the verifier's view is "independent" of the witness used by the prover for proving the statement. In this context, we focus on languages $L \in \mathsf{NP}$ with a corresponding witness relation $\mathbf{R}_L$. Namely, we consider interactions in which on common input x the prover is given a witness in $\mathbf{R}_L(x)$. For any adversarial verifier V^*, let $\mathsf{View}_{V^*}\langle P(w), V(z)\rangle(x)$ be the random variable that denotes V^*'s view in an interaction with P, when V^* is given auxiliary input z, P is given witness w, and both parties are given common input x.

Definition 19 (Witness-indistinguishability)**.** *An interactive protocol (P,V) for $L \in \mathsf{NP}$ is **witness indistinguishable** for $\mathbf{R}_L$ if for every PPT adversarial verifier V^*, and for every two sequences $\{w_x^1\}_{x\in L}$ and $\{w_x^2\}_{x\in L}$, such that $w_x^1, w_x^2 \in R_L(x)$ for every $x \in L$, the following ensembles are computationally indistinguishable over $x \in L$:*

$$\{\mathsf{View}_{V^*}\left\langle P(w_x^1), V^*(z)\right\rangle(x)\}_{x\in L, z\in\{0,1\}^*}$$
$$\approx \{\mathsf{View}_{V^*}\left\langle P(w_x^2), V^*(z)\right\rangle(x)\}_{x\in L, z\in\{0,1\}^*}$$

(The definition of resettable-witness indistinguishability follows that of resettable zero knowledge analogously: witness indistinguishability under a resetting attack as described in Definition 2.)

A.4 Commitment Schemes

Commitment protocols allow a *sender* to commit itself to a value while keeping it secret from the *receiver*; this property is called **hiding**. At a later time, the commitment can only be opened to a single value as determined during the commitment protocol; this property is called **binding**. Commitment schemes come in two different flavors, statistically binding and statistically hiding; we only make use of statistically binding commitments in this paper. Below we sketch the properties of a statistically binding commitment; full definitions can be found in [Gol01].

In statistically binding commitments, the binding property holds against unbounded adversaries, while the hiding property only holds against computationally bounded (non-uniform) adversaries. The statistical-binding property asserts that, with overwhelming probability over the randomness of the receiver, the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that the commitments to any two different values are computationally indistinguishable.

Non-interactive statistically-binding commitment schemes can be constructed using any one-to-one one-way function (see Section 4.4.1 of [Gol01]). Allowing some minimal interaction (in which the receiver first sends a single random initialization message), statistically-binding commitment schemes can be obtained from any one-way function [Nao91, HILL99].

B Construction of an Oracle-Aided UA

In this section, we prove Theorem 7, which is restated below.

Theorem 14 (Theorem 7 restated)**.** *Let SIG' be a canonical sig-com scheme with SIG and Com being its underlying signature scheme and commitment scheme. Then there exists a (SIG', ℓ)-complete $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument with $\ell(n) = 2n$.*

Proof. We construct such a universal argument in Fig. 3, which is essentially identical to the construction of [BG02], except that the Merkle hash tree is replaced by a sig-com tree. Note that both the efficient verification property and the completeness property (with a relatively efficient prover) follow by inspection. Furthermore, note that the (SIG', ℓ)-completeness holds as well, since the prover P only need to access an arbitrary valid (SIG', ℓ) oracle to produce the sig-com tree.

Common Input: An instance $y = (M, x, t)$ of $L_{\mathcal{U}}$; let $n := |y|$.

Auxiliary input to prover: w such that $(y, w) \in \mathbf{R}_{\mathcal{U}}$ holds.

Primitives used:

- A PCP scheme for $L_{\mathcal{U}}$ with auxiliary properties as defined in [BG02], where
  - $P_{\mathrm{PCP}}(y, w)$ generates a PCP proof π for $(y, w) \in \mathbf{R}_{\mathcal{U}}$.
  - V_{PCP} is the non-adaptive verifier for the PCP system, which makes m queries to the PCP proof.
  - $Q_{\mathrm{PCP}}(y, r, i)$ generates the i-th query of V_{PCP} with random tape r, common input y.
- A canonical sig-com scheme SIG' with SIG and Com as the underlying signature and commitment schemes; let $\mathcal{O}^{\mathsf{SIG}}$ be the corresponding signature oracle.

Set Up: Run $(\mathsf{pp}, O) \leftarrow \mathcal{O}^{\mathsf{SIG}}(1^n)$, add pp to common input for P and V. Further, allow P oracle access to O.

Protocol:

P_1 : Generate $\pi \leftarrow P_{\mathrm{PCP}}(y, (w, 1^{t'}))$, where t' is the runtime of M on input (x, w). Use O to generate a sig-com tree for π w.r.t. pp, recording sig-com paths for each leaf. Send (d, l_λ), the depth and the root of the sig-com tree, to V.

V_1 : Uniformly select randomness r for V_{PCP}, and send it to P.

P_2 : Generate queries $\{q_i\}_{i \in m}$ by using $Q_{\mathrm{PCP}}(y, r, i)$ to generate the i-th query for every $i \in [m]$. Generate sig-com paths $\{\vec{\rho_i}\}_{i \in m}$ for the bits $\{b_i = \pi_{q_i}\}_{i \in [m]}$ of π in the sig-com tree. Send the sig-com paths $\{\vec{\rho_i}\}_{i \in m}$ to V.

V accepts when:

- $\mathsf{PATH}^{\mathsf{SIG}'}(\vec{\rho_i}, b_i, q_i, l_\lambda, \mathsf{pp}) = 1$ for every $i \in [m]$.
- V_{PCP} accepts when receiving $\{b_i\}_{i \in [m]}$ as the responses to its oracle queries.

Figure 3: An $\mathcal{O}^{\mathsf{SIG}}$-aided Universal Argument.

It remains to prove the weak proof of knowledge property (for adaptively chosen statements). Our proof is almost identical to that given by Barak and Goldreich in Section 3 of [BG02]. In fact, if we simply replace hash trees with sig-com trees and following their argument exactly, we have the following lemma:

Lemma 15 (implicit in Lemma 3.5 of [BG02])**.** *Let (P, V) be the $\mathcal{O}^{\mathsf{SIG}}$-aided protocol defined in Fig. 3. For every polynomial p, there exist oracle* PPT *algorithms E and* CF *and a polynomial q*

such that for every $n \in \mathbb{N}$ *and every non-uniform* PPT *adversary* P^**, if*

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^n); R \leftarrow \{0,1\}^\infty; y \leftarrow P_R^{*O}(\mathsf{pp}) : (P_R^{*O}(\mathsf{pp}), V(y, \mathsf{pp})) = 1] > 1/p(n),$$

then with probability at least $1/q(n)$ *over* $(\mathsf{pp}, O, R) \leftarrow \mathcal{O}(1^n) \times \{0,1\}^\infty$*, it holds that either*

$$\Pr[r \leftarrow \{0,1\}^\infty; y \leftarrow P_R^{*O}(\mathsf{pp}) : \exists w = w_1, \dots w_t \in \mathbf{R}_{\mathcal{U}}(y) \; s.t. \; \forall i \in [t], E_r^{P_R^{*O}(\mathsf{pp})}(\mathsf{pp}, y, i) = w_i] > \frac{1}{q(n)},$$

or

$$\Pr[(\vec{\rho}_0, \vec{\rho}_1, \gamma, l_\lambda) \leftarrow \mathsf{CF}^{P_R^{*O}(\mathsf{pp})}(\mathsf{pp}); \forall b \in \{0,1\} \; \mathsf{PATH}^{\mathsf{SIG}'}(\vec{\rho}_b, b, \gamma, l_\lambda, \mathsf{pp}) = 1] \geq 1/q(n)$$

Given the above lemma, we observe that, for any P^{*O}, except for finitely many $n \in \mathbb{N}$, the latter condition can only hold with probability at most $1/2q(n)$ over $(\mathsf{pp}, O, R) \leftarrow \mathcal{O}(1^n) \times \{0,1\}^\infty$. Otherwise, we will be able to use $\mathsf{CF}^{P_R^{*O}}$ as an oracle aided adversary that succeeds in breaking the sig-com tree collision resistance of SIG' for infinitely many $n \in \mathbb{N}$ with probability $\geq 1/2(q(n))^2$ over $\mathcal{O}(1^n), R$, and the randomness of CF.

Thus, assuming that SIG' is a secure sig-com scheme, the former condition of the lemma must hold with probability $\geq 1/2q(n)$ over $(\mathsf{pp}, O, R) \leftarrow \mathcal{O}(1^n) \times \{0,1\}^\infty$ for all but finitely many $n \in \mathbb{N}$.

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^n); R, r \leftarrow \{0,1\}^\infty; y \leftarrow P_R^{*O}(\mathsf{pp}) : \exists w = w_1, \dots w_t \in \mathbf{R}_{\mathcal{U}}(y) \; s.t. \; \forall i \in [t], E_r^{P_R^{*O}}(\mathsf{pp}, y, i) = w_i] > \frac{1}{2(q(n))^2}$$

Setting $p'(n) = 2(q(n))^2$, we have that E is a sufficient extractor for the weak proof of knowledge property. □

C Proof of Theorem 8

Here we prove Theorem 8, which is restated below.

Theorem 16 (Theorem 8 restated)**.** *Let* SIG' *be a canonical sig-com scheme with* SIG *and* Com *being its underlying signature scheme and commitment scheme. Then there exists an* $\mathcal{O}^{\mathsf{SIG}}$*-oracle aided argument of knowledge* (P, V) *for* NP*; additionally,*

1. (P, V) *is constant-round and public-coin;*
2. P *does not make any queries to its oracle;*
3. (P, V) *is* (SIG', ℓ)*-oracle-aided zero-knowledge for* $\ell(n) = 2n$*.*

Proof. We show that the construction provided in Fig. 1 and 2 satisfies the desired properties. By inspection, (P, V) is constant-round and public-coin, and P does not make any queries to its oracle. For the (SIG', ℓ)-oracle-aided zero-knowledge property, we construct a simulator identically

to [PR05]. In brief, the straight-line simulator S that will provide a proof to V^* using the second witness for Stage Three, and will use the oracle O to produce a sig-com tree for $\mathtt{ECC}(\Pi)$ in Stage One with $\Pi = V^*_{r_V}$, and also to complete the encrypted UA in Stage Two. We further observe that the ZK simulator will work even with any valid (SIG', ℓ)-oracle, since such an oracle is sufficient to produce a correct sig-com tree in Stage One, and to complete the (SIG', ℓ)-oracle complete UA in Stage Two.

It remains to show the argument of knowledge property. We start by constructing a knowledge extractor E for (P, V). $E(x, \mathsf{pp})$ proceeds as follows: Given oracle access to a malicious prover $P^{*O}(x, \mathsf{pp})$, E internally emulates the role of the honest verifier V for $P^{*O}(x, \mathsf{pp})$ up to the beginning of Stage Three (i.e., the beginning of WI-AOK). Let α denote the partial transcript, and $P^{*O}(x, \mathsf{pp}; \alpha)$ be the "residual" prover. Then E applies the witness extractor E_{WI} for $(P_{\mathsf{WI}}, V_{\mathsf{WI}})$ on $P^{*O}(x, \mathsf{pp}; \alpha)$, and outputs E_{WI}'s output. Note that since $O \leftarrow \mathcal{O}^{\mathsf{SIG}}$ is efficient, $P^{*O}(x, \mathsf{pp}; \alpha)$ is a polynomial size adversary in the plain model.

Clearly by inspection, E runs in expected polynomial time. Let ε be the success probability of P^* in convincing V. We first show that E_{WI} outputs a *valid* witness (either a *true* witness $w \in \mathbf{R}_L(x)$ or a *false* witness $(p_1, p_2, \tau_1, \tau_2) \in \mathbf{R}_{L_2}(c_0, r, c_1, c_2, r', \mathsf{pp})$) with probability $\varepsilon - \mathsf{ngl}(|x|)$.

Let $\varepsilon(\mathsf{pp}, O, \alpha) = \Pr[(P^{*O}, V_{\mathsf{WI}})(x, \mathsf{pp}; \alpha) = 1]$, i.e., the probability that the residual prover $P^{*O}(x, \mathsf{pp}; \alpha)$ convinces V_{WI} in Stage Three. By definition, $\mathbb{E}_{\mathsf{pp},O,\alpha}[\varepsilon(\mathsf{pp}, O, \alpha)] = \varepsilon$. By the argument of knowledge property of the WI-AOK $(P_{\mathsf{WI}}, V_{\mathsf{WI}})$, $E_{\mathsf{WI}}^{P^{*O}(x,\mathsf{pp};\alpha)}$ outputs a valid witness with probability at least $\varepsilon(\mathsf{pp}, O, \alpha) - \mathsf{ngl}(|x|)$. It follows that in the execution of E, E_{WI} outputs a valid witness with probability at least $\mathbb{E}_{\mathsf{pp},O,\alpha}[\varepsilon(\mathsf{pp}, O, \alpha) - \mathsf{ngl}(|x|)] \geq \varepsilon - \mathsf{ngl}(|x|)$.

We proceed to argue that in the execution of E, E_{WI} can only output a false witness with negligible probability. Suppose not, that is, E_{WI} outputs a false witness with some noticeable probability ε'. Then we will use this fact to contradict the collision resistance property of SIG'. We do so in the following two steps:

1. We construct an efficient cheating UA prover P^*_{UA} for $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ that convinces V_{UA} with probability $\mathrm{poly}(\varepsilon(n))$.

2. We use the extractor E_{UA} from the weak argument of knowledge property of $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ together with this P^*_{UA} to build a collision-finder for SIG'.

Step 1: Constructing P^*_{UA}. P^*_{UA} internally emulates P^* and proceeds as follows.

- $P^{*O}_{\mathsf{UA}}(\mathsf{pp})$ runs $c_0 \leftarrow P^{*O}(x, \mathsf{pp})$, samples $r \leftarrow \{0,1\}^n$ and outputs $y = (c_0, r, \mathsf{pp})$ as the adaptively chosen statement.

- $P^{*O}_{\mathsf{UA}}(\mathsf{pp})$ generates the first prover message p_1 as follows: $P^{*O}_{\mathsf{UA}}(\mathsf{pp})$ feeds r to P^{*O}, receives $c_1 \leftarrow P^{*O}(x, \mathsf{pp}; r)$, and continues to emulate the interaction of P^{*O} with an honest V up to the end of Stage Two; let α be the partial transcript and $P^{*O}(x, \mathsf{pp}; \alpha)$ be the "residual" prover. Then P^*_{UA} applies E_{WI} on $P^{*O}(x, \mathsf{pp}; \alpha)$. If E_{WI} outputs a valid $(p_1, p_2, \tau_1, \tau_2) \in \mathbf{R}_{L_2}$, then P^*_{UA} outputs p_1, otherwise, P^*_{UA} aborts.

- Upon receiving r' from V_{UA}, P^*_{UA} rewinds P^* until the point where it awaits the message r', feeds r' to P^{*O}, and receives $c_2 \leftarrow P^{*O}(x, \mathsf{pp}; r)$; let α' denotes the partial transcript. Then P^*_{UA} applies E_{WI} on $P^{*O}(x, \mathsf{pp}; \alpha')$. If E_{WI} outputs a valid $(p'_1, p'_2, \tau'_1, \tau'_2) \in \mathbf{R}_{L_2}$, then P^*_{UA} outputs p'_2, otherwise, P^*_{UA} aborts.

Clearly by inspection, P^*_{UA} runs in expected polynomial time. Furthermore, we can make P^*_{UA} run in strict polynomial time by cutting it off after a certain polynomial time bound with only a small loss in its success probability. It follows by an identical argument to [BG02, PR05] that P^*_{UA} convinces V_{UA} to accept with probability $\mathrm{poly}(\varepsilon')$. Roughly, the argument consists of counting "good" oracles and verifier messages, i.e., those that will let the prover succeeds with "high" probability (see Claim 4.2.1 in [BG02]), together with applying the binding property of the commitment scheme to show that the witnesses extracted by the two executions of E_{WI} have consistent first prover messages (i.e., $p_1 = p'_1$) except with negligible probability (See Lemma A.3 in [PR05]).

Step 2: Finding collision. We now use P^*_{UA} to break the collision resistant property of sig-com tree corresponding to SIG', which contradicts Lemma 6. Let $1/p$ be a lower bound on the success probability of P^*_{UA} for some polynomial p. Let E_{UA} be the corresponding weak knowledge extractor for $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$. Recall the weak argument of knowledge property guarantees that

$$\Pr[\mathsf{pp}, O \leftarrow \mathcal{O}(1^n); \omega, \nu \leftarrow \{0,1\}^\infty; y \leftarrow P^{*O}_{\mathsf{UA},\omega}(\mathsf{pp}) : \exists w = w_1, \dots w_t \in \mathbf{R}_{L_1}(y) \; s.t. \\ \forall i \in [t], E^{P^{*O}_{\mathsf{UA},\omega}}_{\mathsf{UA},\nu}(\mathsf{pp}, y, i) = w_i] > \frac{1}{p'(n)},$$

where ω, ν denote the random tapes of P^*_{UA} and E_{UA}, respectively, p' is some polynomial, and the witness w is of the form $(\tau_0, d, l_\lambda, C, \{\vec{\rho}_i\}_{i\in[2^d]})$. To simplify the notation, let $\mathsf{suc}(pp, O, \nu, \omega) = 1$ if the extraction successfully extracts a valid witness $w \in \mathbf{R}_{L_1}(y)$.

We construct a PPT adversary A that breaks the collision resistance property of sig-com tree corresponding to SIG'. A on input 1^n and vk with oracle access to a signing oracle $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$ proceeds as follows.

- A uses vk and its signing oracle to emulate an $\mathcal{O}^{\mathsf{SIG}}$ oracle with $\mathsf{pp} = \mathsf{vk}$ and $O = \mathsf{Sign}_{\mathsf{sk}}(\cdot)$.
- A samples $\omega, \tilde{\omega}$, and let $y \leftarrow P^{*O}_{\mathsf{UA},\omega}(\mathsf{pp}), \tilde{y} \leftarrow P^{*O}_{\mathsf{UA},\tilde{\omega}}(\mathsf{pp})$; recall from our definition of P^*_{UA} that $y = (c_0, r, \mathsf{pp}), \tilde{y} = (c_0, \tilde{r}, \mathsf{pp})$ will each contain the same c_0 and pp components, while r and $\tilde{r}$ are selected independently uniformly at random.
- A samples $\nu, \tilde{\nu}$, and applies $E^{P^{*O}_{\mathsf{UA},\omega}}_{\mathsf{UA},\nu}(\mathsf{pp}, y, \cdot)$ and $E^{P^{*O}_{\mathsf{UA},\tilde{\omega}}}_{\mathsf{UA},\tilde{\nu}}(\mathsf{pp}, \tilde{y}, \cdot)$ to extract (part of) witnesses $w = (\tau_0, d, l_\lambda, C, \{\vec{\rho}_i\}_{i\in[2^d]})$ and $\tilde{w} = (\tilde{\tau}_0, \tilde{d}, \tilde{l}_\lambda, \tilde{C}, \{\vec{\tilde{\rho}}_i\}_{i\in[2^d]})$ as follows: (1) A first extracts (d, l_λ) and $(\tilde{d}, \tilde{l}_\lambda)$. A aborts if the extraction fails at any point. (Note that by the binding property of the commitment, $(d, l_\lambda) = (\tilde{d}, \tilde{l}_\lambda)$ except with negligible probability.) (2) Then A samples $i \leftarrow [2^d]$, and extracts $(C_i, \vec{\rho}_i)$ and $(\tilde{C}_i, \vec{\tilde{\rho}}_i)$
- If $C_i \neq \tilde{C}_i$, then A outputs $(\vec{\rho}_i, \vec{\tilde{\rho}}_i, i, l_\lambda)$ if $C_i = 0$, and $(\vec{\tilde{\rho}}_i, \vec{\rho}_i, i, l_\lambda)$ if $C_i = 1$.

We now show that A can breaks the collision resistance property with non-negligible probability. Note that A runs the knowledge extraction twice with respect to the same pp, O but independent copies of (ν, ω) and $(\tilde{\nu}, \tilde{\omega})$. Recall that the extraction succeeds with probability at least $1/p'$. For at least $1/(2p')$-fraction of (pp, O), it holds that $\Pr[\mathsf{suc}(\mathsf{pp}, O, \nu, \omega) = 1 | (\mathsf{pp}, O)] \geq 1/(2p')$. Therefore, with probability at least $(1/2p')^3$ over $(\mathsf{pp}, O, \nu, \omega, \tilde{\nu}, \tilde{\omega})$, both $\mathsf{suc}(\mathsf{pp}, O, \nu, \omega) = \mathsf{suc}(\mathsf{pp}, O, \tilde{\nu}, \tilde{\omega}) = 1$, i.e., both extractions invoked by A succeed.

Finally, we note that the independently drawn r and $\tilde{r}$ are different with overwhelming probability, and so the Π and $\tilde{\Pi}$ underlying C and $\tilde{C}$ will also be different with overwhelming probability.

Since we used an error-correcting code $\mathtt{ECC}$ with constant min-distance, if $\Pi \neq \tilde{\Pi}$, then for a randomly chosen i, $C_i \neq \tilde{C}_i$ with constant probability c, meaning the two paths outputted by A will have different leaf labels. Thus with probability $\geq c/(2p')^3$, A successfully outputs a pair of colliding paths with the same root but different leaf labels, breaking the collision resistance of sig-com trees corresponding to SIG'. $\square$