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

Kai-Min Chung
Cornell University
chung@cs.cornell.edu

Rafael Pass[*]
Cornell University
rafael@cs.cornell.edu

Karn Seth
Cornell University
karn@cs.cornell.edu

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.

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.

Categories and Subject Descriptors

F.1.2 [**Theory of Computation**]: Interactive and reactive computation

[*]Pass is supported in part by an Alfred P. Sloan Fellowship, a Microsoft Research Faculty Fellowship, NSF Awards CNS-1217821 and CCF-1214844, NSF CAREER Award CCF-0746990, 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.

General Terms

Theory

Keywords

zero-knowledge, non-black-box simulation, one-way functions, resettable security

1. INTRODUCTION

Zero-knowledge (ZK) interactive protocols [16] 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* [15]: 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. [13, 10, 3, 26].

In a breakthrough result from 2001, Barak [1] 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 [13] such protocols cannot be proved zero-knowledge using just black-box simulation. In the same year, Barak, Goldwasser, Goldreich and Lindell [3] 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 crucial 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. [2, 24, 25, 11]).

One important limitation of the non-black-box simulation technique of Barak [1] (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 [2]) 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 (essentially) necessary [14, 17, 23], 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 that cannot be proven secure using black-box simulation techniques)?*

A very recent elegant work by Bitansky and Paneth [6] 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 need for CRHs in the construction of [11] 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.

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.

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* [19].

Roughly speaking, on common input 1^n and $x \in \{0,1\}^{\mathrm{poly}(n)}$, 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) [20, 2] 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

[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 [2].
[2]Most 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).

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. [3] 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 [27]. 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 [12]. We observe that such signature scheme share a lot of properties with CRHs. First of 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 [18] 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 [25], 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 [1, 25].

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 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 [18], 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

[3]Strictly speaking, Barak's protocol is not a argument of knowledge, but rather a "weak" argument of knowledge (see [2, 3] for more details), but the transformation of [3] applies also to such protocol.

[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.

[3] 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 [13, 3] 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 [8, 3] with some minor modifications.

To achieve simulateously resettable zero-knowledge, we instead instantiate the protocol of Deng, Goyal and Sahai [11] with signature trees, in exactly the same way as Barak's protocol. Resettable-soundness follows exactly as in [11], relying on the collision-resistance property of signature trees. Resettable-zero-knowledge is more tricky though: [11] provides an intricate simulation strategy that combines black-box simulation, using rewinding, and non-black-box simulation (as in [1]). 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 [11] 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 [11] but instead rewind each slot (that was being rewound once in [11]) $3g(n)$ times. It follows using a slight generalization of the argument in [11] 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 [11]. 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 [11] 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 [6] does not rely on PCPs; on the other hand, it relies on some other quite heavy machinery: "unobfuscatable functions" [4] and general secure two-party computation [14].

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., [7].) 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 [25] variant of [1]) 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 $M(c) = r$ in a *piecemeal* fashion, by making the verifier certify every step of the computation of M. This strategy is very similar to one employed in the "impossibility of instantiating random oracles" result of [9][6] (On a high-level, this type of piecemeal decomposition is also somewhat similar to what is done in the impossibility result of [4]; as such our approach brings out the connection between the techniques from [1] and [6].) 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 tree hash[7] of a *current-configuration*, a commitment c_2 to a tree-hash of a *next-configuration*, and a witness indistinguishable argument of knowledge that either a) $x \in L$ or b) *next-configuration* is a starting configuration or c) performing *one step* of computation given *current-configuration* leads to *next-configuration*, and *current-configuration* has been previously signed. (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_2. Roughly speaking, the above "slot" makes it possible for the simulator to get a signature on (commitments to signature-trees of) 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_t in the computation of $M(\sigma)$ (using condition c). Thus, finally, the simulator can get a signature on 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 appears in the full version of this paper.

1.3 Subsequent Work

A very recent elegant work by Bitansky and Paneth [5] (developed subsequently to our results) shows an alternative approach for obtaining resettably-sound arguments (and re-

[6]They key difference is that construction of [9] 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.

lated primitives) from one-way functions, by first constructing functions that are "approximately" unobfuscatable, and relying on the connection between resettable-soundness and unobfuscatable functions from [6].

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 [2] 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 [25] 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 [3] to this protocol to obtain an oracle-aided resettably-sound zero-knowledge argument of knowledge satisfying the same key property (the results of [3] 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.

Applications (such as simultanously resettable zero-knowledge) are presented in the full version of the paper.

2. DEFINITIONS

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

We start by recalling the definition of zero knowledge from [16].

Definition 1 (Zero-knowledge [16]). *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 $x \in L$:*

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

Let us recall the definition of resettable soundness due to [3].

Definition 2 (Resettably-sound Arguments [3]). *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 $r_1, \dots, r_t$, 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).

Definition 3 (Fixed-input RS Arguments [26]). *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 (which essentially follows from techniques in [3]) 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 $L \in \mathsf{NP}$. Then there exists a protocol (P', V') that is a (full-fledged) resettably-sound zero-knowledge (resp. witness indistinguishable) argument of knowledge for L.*

The proof is found in the full version.

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 do not 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 in the same way.

[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 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* $n \in \mathbb{N}, m \in \{0,1\}^*$,

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

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

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

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

Strong, length-ℓ, deterministic signature schemes with $\ell(n) = n$ are known based on the existence of OWFs; see [22, 27, 12] for further details. In the rest of this paper, whenever we refer to signature schemes, we always means strong, length-n deterministic signature schemes.

Let us first note that strong 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 $n \in \mathbb{N}$,*

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

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 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 $\ge \frac{1}{p(n)}$, we forge a signature σ for some m before ever querying the signing oracle and receiving σ as a response. $\square$

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 the root l_λ for the bit b at leaf $\gamma \in \{0,1\}^d$ is a vector* $\vec{\rho} = ((l_0, l_1), ((l_{\gamma_{\le 1}0}, l_{\gamma_{\le 1}1}), \ldots (l_{\gamma_{\le d-1}0}, l_{\gamma_{\le d-1}0}))$ *such that for every* $i \in \{0, \ldots, d-1\}$, $\mathsf{Ver}_{\mathsf{vk}}((l_{\gamma_{\le i}0}, l_{\gamma_{\le i}1}), l_{\gamma_{\le i}}) = 1$. *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:*

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

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. $\square$

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 [21]), by simply adding the first message q to the verfication 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 a 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 the root l_λ for the 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:*

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

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 RS-ZK

In this section we show how to construct a resettably-sound ZK argument in an oracle-aided model where the 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 Arg). *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 $x \notin L$,*

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

We will also define an $\mathcal{O}$-oracle aided version of arguments of knowledge, essentially analogously to their canonical definitions, except with a setup phase in which pp and O are generated and made available to the players. The formal definitions can be found in the full version of this paper.

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 [2] and closely related to CS-proofs [20]) 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 presumably "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 [2] definition of universal arguments.

Definition 11 (Oracle-aided Universal argument). *An oracle-aided interactive argument (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 by 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 [2] 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 [2] and their analysis will be useful to us. Recall that in the construction of [2] tree hashing is used to hash down a "long" PCP proof into a fixed-length "tree root"; the soundness property relies on collision resistance 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 [2] 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 12 (Signature Oracle). *A signature oracle $\mathcal{O}^{\mathsf{SIG}}$ is defined 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 13 (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 commitment are also considered *valid* sig-com oracles. (These are precisely the kind of oracles we will be forced to use later on).

Definition 14. *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 polynomial ℓ and a (SIG', ℓ)-complete $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument Π.*

The proof of the theorem identically follows that of Barak and Goldreich [2], with a minor modification to deal with adaptively chosen statements when proving the weak argument of knowledge property. The proof is found in the full version.

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 15 (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 non-uniform adversarial verifier V^*, there exists a simulator S, such that for every valid (SIG', ℓ) oracle $\mathcal{O}'$, the following ensembles are indistinguishable over $x \in L$,*
$$\{\mathsf{pp}, O \leftarrow \mathcal{O}'(1^{|x|}); \mathsf{View}_{V^*}(P(w), V^*(z))(x, \mathsf{pp})\}_{x,w,z} \approx \{\mathsf{pp}, O \leftarrow \mathcal{O}'(1^{|x|}); S^O(x, z, \mathsf{pp})\}_{x,w,z}$$
where the ensembles are over $x \in L, w \in R_L(x), z \in \{0,1\}^*$.

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

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 [1] with sig-com trees, and then apply the PRF transformation of [3] 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 [25], 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^{O}_{ZK} sends a commitment $c = \mathsf{Com}(0^n)$, and then V_{ZK} sends back a challenge $r \in \{0,1\}^{2n}$ as in Barak's protocol. In Stage 2, P^{O}_{ZK} and V_{ZK} first execute an "encrypted" universal argument $(P^{O}_{\mathsf{UA}}, V_{\mathsf{UA}})$ of the statement that *"c is a commitment to a sig-com tree root of a program M and there is a short string $y \in \{0,1\}^n$ such that $M(y) = r$,"* 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^{O}_{ZK} and V_{ZK} 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^{O}_{ZK} can convince the verifier by proving $x \in L$ in the final witness indistinguishable argument without making any oracle queries.

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 $\ell(n) = 2n$.*

The proof of Theorem 8 is found in the full version. The proof closely follows [1, 25] 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]

Finally, we apply the PRF transformation of [3] to $(P^{O}_{\mathsf{ZK}}, V_{\mathsf{ZK}})$ to achieve resettable soundness. More precisely, we modify the public-coin verifier V_{ZK} to a "PRF-verifier" $\tilde{V}_{\mathsf{ZK}}$ 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 [3] relativizes and as a consequence we have the following theorem:

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

[11]In [1, 25] these issues do 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, $\mathcal{O}^{\mathsf{SIG}}$ defined relative to SIG, and a $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument $(P_{\mathsf{UA}}, V_{\mathsf{UA}})$ defined in Sec. 4.1.

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.

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 OA-UA):

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_{OA-UA}

P_3: Send $c_2 = \mathsf{Com}(0^k, \tau_2)$ for uniformly selected τ_2, where k is the length of P_{OA-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 strong error-correcting code. 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 schemes SIG' and $\mathtt{ECC}$. Let (P_{UA}, V_{UA}) be the $\mathcal{O}^{\mathsf{SIG}}$-aided universal argument constructed in Sec. 4.1. 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_{UA}, V_{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 $\mathcal{O}^{\mathsf{SIG}}$-aided ZK protocol.

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. RS-ZK 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 a $\mathcal{O}^{\mathsf{SIG}}$-aided resettably sound argument of knowledge for the 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 $(P(w), V)(x, \mathsf{pp})$ with $\mathsf{pp} = \mathsf{vk}$.

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

Note that here we only obtain a *single-instance* resettably sound argument of knowledge (defined in Defn. 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. The proof of Lemma 10 is found in the full version. We provide a very brief sketch below.

PROOF. (sketch) Completeness of $(\tilde{P}, \tilde{V})$ follows directly from the completeness of (P, V) since by assumption, P never makes any oracle queries. Resettable-soundness and the argument of knowledge property, roughly speaking, follow by emulating all signature slot messages using the oracle; note that we here rely on the fact that the signature scheme is deterministic to ensure that "rewindings" of the signature slot can be emulated as oracle queries. 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 [13], we need to take special care to deal with verifier's that only provide valid signatures with very small probability. □

Acknowledgements

We are very grateful to Ran Canetti for pointing out the connection to [9].

6. REFERENCES

[1] B. Barak. How to go beyond the black-box simulation barrier. In *FOCS '01*, pages 106–115, 2001.
[2] B. Barak and O. Goldreich. Universal arguments and their applications. In *Computational Complexity*, pages 162–171, 2002.
[3] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS'02*, pages 116–125, 2001.
[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
[5] N. Bitansky and O. Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, 2011.
[6] N. Bitansky and O. Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *FOCS*, 2012.
[7] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Advances in Cryptology*, pages 93–118, 2001.
[8] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge (extended abstract). In *STOC '00*, pages 235–244, 2000.
[9] R. Canetti, O. Goldreich, and S. Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. In *TCC*, pages 40–57, 2004.
[10] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC '01*, pages 570–579, 2001.
[11] 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.
[12] O. Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.
[13] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.
[14] O. Goldreich, S. Micali, and A. 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.
[15] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994.
[16] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.
[17] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.
[18] H. Lin and R. Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, pages 705–714, 2011.
[19] R. Merkle. Digital signature system and method based on a conventional encryption function, Nov. 14 1989. US Patent 4,881,264.
[20] S. Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.
[21] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
[22] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *STOC '89*, pages 33–43, 1989.
[23] R. Ostrovsky and A. Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *Theory and Computing Systems, 1993*, pages 3–17, 1993.
[24] R. Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC '04*, pages 232–241, 2004.
[25] R. Pass and A. Rosen. New and improved constructions of non-malleable cryptographic protocols. In *STOC '05*, pages 533–542, 2005.
[26] R. Pass, W.-L. D. Tseng, and D. Wikström. On the composition of public-coin zero-knowledge protocols. *SIAM J. Comput.*, 40(6):1529–1553, 2011.
[27] J. Rompel. One-way functions are necessary and sufficient for secure signatures, 1990.