Predictable Arguments of Knowledge

Antonio Faonio[1], Jesper Buus Nielsen[1], and Daniele Venturi[2]

[1] *Department of Computer Science, Aarhus University, Denmark* ,
{antfa,jbn}@cs.au.dk

[2] *Department of Information Engineering and Computer Science, University of Trento* ,
daniele.venturi@unitn.it

January 11, 2017

Abstract

We initiate a formal investigation on the power of *predictability* for argument of knowledge systems for *NP*. Specifically, we consider private-coin argument systems where the answer of the prover can be predicted, given the private randomness of the verifier; we call such protocols Predictable Arguments of Knowledge (PAoK).

Our study encompasses a full characterization of PAoK, showing that such arguments can be made extremely laconic, with the prover sending a single bit, and assumed to have only one round (i.e., two messages) of communication without loss of generality.

We additionally explore PAoK satisfying additional properties (including zero-knowledge and the possibility of re-using the same challenge across multiple executions with the prover), present several constructions of PAoK relying on different cryptographic tools, and discuss applications to cryptography.

Acknowledgements

The first author and second author acknowledge support by European Research Council Starting Grant 279447. The first author and second author acknowledge support from the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation.

Contents

1 Introduction

Consider the classical proof system for Graphs Non-Isomorphism where, on common input two graphs (G_0, G_1), the verifier chooses a random bit b, and sends a uniformly random permutation of the graph G_b to the prover. If the two graphs are not isomorphic the prover replies correctly sending back the value b.

A peculiar property of the above proof system is that the verifier knows in advance the answer of the prover, i.e., the answer given by the prover is *predictable.* Another property is that it uses only one round of communication and that the prover sends a single bit. Following the work of Goldreich *et al.* [GVW02] we call a proof system with these properties *extremely laconic.*

In this paper, we study the notion of predictability in interactive proof systems for NP. More specifically, we focus on the cryptographic setting where the prover's strategy is efficiently computable and, moreover, we aim for the notion of knowledge soundness, where any convincing polynomial-time prover must "know" the witness relative to the instance being proven.

We formalize this notion of Predictable Arguments of Knowledge (PAoK), explore their properties and applications, and provide several constructions based on various cryptographic tools and assumptions.

1.1 Our Contributions and Techniques

We proceed to describe our results and techniques in more details.

Characterizing PAoK. Syntactically a PAoK is a multi-round protocol $(\mathcal{P}, \mathcal{V})$ where in each round: (i) The verifier $\mathcal{V}$, given the instance x and private coins r, generates a challenge c (that is sent to $\mathcal{P}$) together with a predicted answer b; (ii) The prover $\mathcal{P}$, given (x, w, c), generates an answer a. The prover is said to convince the verifier if and only if $a = b$ in all rounds.

Apart from being complete—meaning that an honest prover convinces the verifier with overwhelming probability—PAoK satisfy the standard property of *knowledge soundness.* Informally, this means that given any successful prover convincing the verifier on instance x with probability ϵ, there exists an efficient extractor recovering a witness for x with probability polynomially related to ϵ. Looking ahead, our definition of knowledge soundness is parametrized by a so-called instance sampler. Intuitively this means that only instances sampled through the sampler are extractable, and allows to consider more fine-grained flavours of extractability.[1]

Our first result is that PAoK can always be made extremely laconic, both in term of round complexity and of message complexity (i.e., the number of bits sent by the prover). Such a characterization is obtained as follows:

- First, we show that one can collapse any multi-round PAoK into a one-round PAoK with higher message complexity. Let $(\mathcal{P}, \mathcal{V})$ be a ρ-round PAoK, where $\mathcal{V}$ generates several challenges $(c_1, \ldots, c_\rho)$ with c_i used during round i.[2] We turn $(\mathcal{P}, \mathcal{V})$ into a one-round predictable argument $(\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ where the multi-round PAoK is "cut" at a random index $i^* \in [\rho]$; this essentially means that $\tilde{\mathcal{V}}$ runs $\mathcal{V}$ and forwards $(c_1, \ldots, c_{i^*})$, whereas $\tilde{\mathcal{P}}$ runs $\mathcal{P}$ and replies with $(a_1, \ldots, a_{i^*})$. One can show that, if the initial PAoK has knowledge error ϵ, the transformed PAoK has knowledge error ϵ/ρ. The latter can finally be made negligible via parallel repetition. It is important to notice that parallel repetition, in general, does not amplify soundness for argument systems [BIN97, PW12]. However, it is well known that for secret-coin one-round arguments (such as PAoK), parallel repetition amplifies (knowledge) soundness at an exponential rate [BIN97].
- Second, we show how to reduce the prover's answer length to a single bit[3] as follows. Let $(\mathcal{P}, \mathcal{V})$ be a PAoK with ℓ-bit answers. We define a new PAoK $(\mathcal{P}', \mathcal{V}')$ where the verifier $\mathcal{V}'$ runs $\mathcal{V}$ in order to obtain a pair (c, b), samples randomness r, and defines the new predicted answer to be the inner product between b and r. Given challenge (c, r) the prover $\mathcal{P}'$ simply runs $\mathcal{P}$ in order to obtain a and defines the answer to be the inner product between a and r. Knowledge soundness follows by the Goldreich-Levin hard-core bit theorem [GL89].

Interestingly, we can wrap up the two results together showing that any PAoK, no matter of the round or message complexity, can be made extremely laconic.

Constructions. Next, we turn to constructing PAoK. Our starting point is the observation that full-fledged PAoK for a relation R imply (and in fact are equivalent to) extractable witness encryption [GKP$^+$13] (Ext-WE) for the same relation R. Briefly, a witness encryption scheme allows to encrypt an arbitrary message using a statement x belonging to an *NP*-language L; decryption can be performed by anyone knowing a valid witness w for x. Extractable security means that from any adversary breaking semantic security of the encryption scheme, we can obtain an extractor computing a valid witness for x.

The equivalence between PAoK and Ext-WE can be seen as follows:

- From Ext-WE to PAoK we encrypt a random bit a using the encryption scheme and then ask the prover to return a.

[1] Similar fine-grained definitions have already been considered in the literature, e.g., for differing-inputs obfuscation [BST14].

[2] It is easy to see that generating all the challenges at the same time, independently of the prover's answers, is without loss of generality.

[3] This further justifies our interest to arguments (as opposed to *proofs*) for *NP* as Goldreich *et al.* [GVW02] showed that unless the polynomial-time hierarchy collapses there does not exist a laconic proof system for all *NP*.

- From PAoK to Ext-WE, we first make the PAoK extremely laconic, then we generate a challenge/answer pair (c, a) for the PAoK, and encrypt a single bit β as $(c, a \oplus \beta)$.[4]

In light of the recent work by Garg *et al.* [GGHW14], the above result can be seen as a negative result. In particular, [GGHW14] shows that, under the conjecture that a certain special-purpose obfuscator exists, it is impossible to have an Ext-WE scheme for a specific *NP* relation. The reason for this depends on the auxiliary information that an adversary might have on the input: The assumed special-purpose obfuscator could be used to obfuscate the auxiliary input in a way that allows to decrypt ciphertexts, without revealing any information about the witness. As stated in [GGHW14], such a negative result can be interpreted as an "implausibility result" on the existence of Ext-WE with arbitrary auxiliary input for all of *NP*. Given the equivalence between PAoK and Ext-WE such an implausibility result carries over to PAoK as well.[5]

Motivated by the above discussion, we propose two constructions of PAoK that circumvent the implausibility result of [GGHW14] by either restricting to specific *NP* relations, or by focusing on PAoK where knowledge soundness is only required to hold for a specific class of instance samplers (and thus for restricted auxiliary inputs). More in details:

- We show a simple connection between PAoK and so-called Extractable Hash-Proof Systems[6] [Wee10] (Ext-HPS): Given an Ext-HPS for a relation R it is possible to construct a PAoK for a related relation R' in a natural way.
- We can construct a PAoK for a specific instance sampler by assuming a weak[7] form of differing-inputs obfuscation. The challenge c corresponds to an obfuscation of the circuit that hard-wires the instance x and a random value b, and upon input w returns b if and only if (x, w) is in the relation.

Interestingly, we can show that, for the special case of so-called random self-reducible relations,[8] a PAoK with knowledge soundness w.r.t. the instance sampler that corresponds to the algorithm for re-randomizing an instance in the language, can be generically leveraged to obtain a full-fledged PAoK (with arbitrary auxiliary input) for any *NP*-relation that is random-self reducible.

Zero-Knowledge PAoK. Notice that, as opposed to standard arguments, predictable arguments are non-trivial to construct even without requiring them to be zero-knowledge (or even witness indistinguishable).[9] Nevertheless, it is possible (and interesting) to consider PAoK that additionally satisfy the zero-knowledge property. It is well known that argument systems with a deterministic prover, such as PAoK, cannot be zero-knowledge in the plain model [GO94]. Motivated by this, given any PAoK (for some fixed relation), we propose two different transformations to obtain a zero-knowledge PAoK (for the same relation):

- The first transformation is in the non-programmable random oracle model. Here we exploit the fact that PAoK are honest-verifier zero-knowledge. Our strategy is to force the malicious verifier to act honestly; we achieve this by having the prover check that the

[4] Domain extension for Ext-WE can be obtained by encrypting each bit of a message individually.

[5] Very recently, Bellare *et al.* [BSW16] show that assuming sub-exponential one-way functions and sub-exponential indistinguishability obfuscation, differing-input obfuscation for Turing Machines [ABG$^+$13] is impossible. While this result adds another negative evidence, it does not apply directly to Ext-WE.

[6] The connection between Hash Proof Systems and Witness Encryption was already noted by [GGHW14].

[7] Namely, following the terminology in [BST14], extractability only holds for a specific class of circuit samplers, related to the underlying instance sampler.

[8] Roughly speaking, a random self-reducible relation is a relation for which average-case hardness implies worst-case hardness.

[9] This is because the trivial protocol where the prover forwards a witness is not predictable.

challenge was honestly generated using randomness provided by the random oracle. In case the check fails the prover will not reveal the answer, but instead it will output a special symbol $\perp$. To ensure knowledge soundness we define the check to be dependent on the prover's message, in such a way that a malicious prover cannot obtain the (private) randomness of the verifier in case it does not already know the correct answer.
- The second transformation is in the common random string (CRS) model, and works as follows. The verifier sends the challenge c together with a non-interactive zero-knowledge proof π that c is "well formed" (i.e., there exists random coins r such that the verifier of the underlying PAoK with coins r returns a pair (c, b)).

We leave it as an interesting open problem to construct a witness indistinguishable PAoK in the plain model.

Predictable ZAP. In the basic definition of PAoK, the verifier generates the challenge c (together with the predicted answer b) depending on the instance x being proven. We also look at the special case where the challenge is generated in an instance-independent manner, together with a trapdoor that later allows to predict the prover's answer a. The goal here is to have the *same* challenge being used across multiple executions of a PAoK with the prover.

Protocols of this type have been already considered in the literature under the name of ZAP [DN07]. There are however a few crucial differences: (i) ZAP are public-coin, whereas predictable arguments are secret-coin; (ii) ZAP are witness indistinguishable, whereas predictable arguments are interesting even without requiring such a property. Hence, we formalize the notion of Predictable ZAP (PZAP) which is a kind of secret-coin ZAP in which the prover's answer can be predicted (given the secret coins of the verifier and some trapdoor), and the same challenge can be re-used across multiple executions. We insist on PZAP satisfying knowledge soundness, but we do not require them to be witness indistinguishable; the definition of knowledge soundness features a malicious prover that can adaptively choose the target instance while keeping oracle access to the verifier algorithm. We also consider a weaker flavour, where the prover has no access to the verifier.

We give a construction of PZAP relying on the recently introduced tool of Extractable Witness PRF [Zha16]. We also show that weak PZAP can be generically leveraged to PZAP using standard cryptographic tools. This result shows that, under some standard cryptographic assumptions, for any construction of weak PZAP there exists *another* construction satisfying the definition of PZAP. It is interesting to understand if given a construction of weak PZAP the construction itself already satisfies the definition of PZAP. We give a negative evidence for this question. Namely, we show a black-box separation between weak PZAP and PZAP, ruling out a large class of black-box reductions from the former to the latter.

Applications. Although we find the concept of PAoK to be interesting in its own right, we also discuss applications of PAoK to proving lower bounds in two different cryptographic settings:

- Leakage-tolerant interactive protocols (as introduced by Bitanski, Canetti and Halevi [BCH12]) are interactive protocols whose security degrades gracefully in the presence of arbitrary leakage on the state of the players.

  Previous work [NVZ13] showed that any leakage-tolerant interactive protocol for secure message transmission, tolerating leakage of poly-logarithmic size on the state of the receiver, needs to have secret keys which are as long as the total number of bits transmitted using that key. Using PAoK, we can strengthen this negative result to hold already for leakage of a constant number of bits. Details are deferred in Appendix A.

- Non-malleable codes (as introduced by Dziembowski *et al.* [DPW10]) allow to encode a message in such a way that the decoding of a tampered codeword either yields the original message or a completely unrelated value.
  Previous work [FMNV15] showed an interesting application of non-malleable codes to protecting arbitrary computation (carried out by a von Neumann architecture) against tampering attacks. This result requires to assume a leakage- and tamper-free CPU which is used to carry out "simple" operations on a constant number of encodings.
  A natural idea to weaken the assumption of a leakage-proof CPU, would be to design a code which remains non-malleable even given a small amount of leakage on the encoded message. Subsequent to our work [FN15], the concept of PAoK has been exploited to show that such non-malleable codes tolerating leakage from the encoding process cannot exist (under the assumption that collision-resistant hash functions exist).

1.2 Giving up on Knowledge Extraction

As already discussed above, the implausibility result of Garg *et al.* [GGHW14] has negative implications on some of our results. We were able to circumvent these implications by either constructing PAoK for restricted relations, or by considering weaker flavours of extractability. Yet another way to circumvent the implausibility result of [GGHW14] is to give up on knowledge soundness and to consider instead standard computational soundness (i.e., a computationally bounded malicious prover cannot convince the verifier into accepting a false statement).

Let us call a multi-round, predictable, computationally sound interactive protocol a *predictable argument*. It is easy to see that all our results for PAoK continue to hold for predictable arguments. In particular: (i) Predictable arguments can be assumed w.l.o.g. to be extremely laconic; (ii) There exists a predictable argument for a relation R if and only if there exists a (non-extractable) witness encryption scheme for R; (iii) We can construct a predictable argument for a relation R given any hash-proof system for R;[10] (iv) Computationally sound PZAP can be obtained based on any (non-extractable) Witness PRF.

1.3 Additional Related Work

A study of interactive proofs with laconic provers was done already in [GH98, GVW02]. They did not investigate proofs of *knowledge*, though. As explained above our notion of PAoK is intimately related to extractable witness encryption, as first proposed by Goldwasser *et al.* [GKP$^+$13]—where it is argued that the construction of Garg *et al.* [GGSW13] is extractable. See [AFP15, DS15] for more recent work on witness encryption.

In [GOVW12], Garg *et al.* introduce the concept of Efficiently Extractable Non-Interactive Istance-Dependent Commitment Scheme (Ext-NI-ID Commitment for short). The primitive resembles the concept of PAoK, however there is a crucial difference. Ext-NI-ID Commitments are statistical hiding, this implies that an Ext-NI-ID can be used to construct a Predictable Argument with "statistical soundness" for the same language, however, the reverse implication does not hold.

The problem we faced to amplify knowledge soundness of PAoK shares similarities with the problem of amplifying computational soundness for argument systems. Although it is well known that parallel repetition does not work in general [BIN97, PW12], there are some exceptions such as 3-message arguments [BIN97, CHS05], public-coin arguments [PV07, CP15], and simulatable

[10] We note that, in the other direction, predictable arguments seem to imply some kind of hash-proof system where "statistical smoothness" is replaced by "computational smoothness." We leave it as an interesting direction for future research to explore potential applications of such "computationally smooth" hash-proof systems and their connection to trapdoor hash-proof system (see Benhamouda *et al.* [BBC$^+$13]).

arguments [HPWP10, CL10] (a generalization of both 3-message and public-coin). Relevant to ours is the work of Haitner on random-terminating arguments [Hai13].

1.4 Roadmap

We start by setting some basic notation, in Section 2. The definition of PAoK, together with their characterization in terms of round-complexity and amount of prover communication, can be found in Section 3. In Section 4 we prove the equivalence between PAoK and extractable witness encryption, and we further explore constructions of PAoK for random self-reducible relations and for relations admitting an extractable hash-proof system. The two compilers yielding zero-knowledge PAoK in the CRS model and in the non-programmable random oracle model are presented in Section 5. In Section 6 we give the definition of (weak) predictable ZAP, and exhibit a construction of this primitive from any extractable witness PRF.Finally, in Section 7, we discuss a few interesting open problems related to our work.

2 Preliminaries

2.1 Notation

For $a, b \in \mathbb{R}$, we let $[a, b] = \{x \in \mathbb{R} \ : \ a \leq x \leq b\}$; for $a \in \mathbb{N}$ we let $[a] = \{1, 2, \ldots, a\}$. If x is a string, we denote its length by $|x|$; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the number of elements in $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\!\$\, \mathcal{X}$. When $\mathcal{A}$ is an algorithm, we write $y \leftarrow\!\!\$\, \mathcal{A}(x)$ to denote a run of $\mathcal{A}$ on input x and output y; if $\mathcal{A}$ is randomized, then y is a random variable and $\mathcal{A}(x; r)$ denotes a run of $\mathcal{A}$ on input x and randomness r. An algorithm $\mathcal{A}$ is *probabilistic polynomial-time* (PPT) if $\mathcal{A}$ is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathcal{A}(x; r)$ terminates in at most $poly(|x|)$ steps. Vectors and matrices are typeset in boldface. For a vector $\mathbf{v} = (v_1, \ldots, v_n)$ we sometimes write $\mathbf{v}[i]$ for the i-th element of $\mathbf{v}$. We use Maj to denote the majority function.

Throughout the paper we let $\kappa \in \mathbb{N}$ denote the security parameter. We say that a function $\nu : \mathbb{N} \to [0, 1]$ is negligible in the security parameter, if $\nu(\kappa) = \kappa^{-\omega(1)}$. A function $\mu : \mathbb{N} \to [0, 1]$ is noticeable in the security parameter, if there exists a positive polynomial $p(\cdot)$ such that $\nu(\kappa) \geqslant 1/p(\kappa)$ for infinitely many $\kappa \geqslant \kappa_0$.

Let X and Y be a pair of random variables. The statistical distance between X and Y is defined as $\Delta(X, Y) := \max_{\mathcal{D}} |\Pr[\mathcal{D}(X) = 1] - \Pr[\mathcal{D}(Y) = 1]|$, where the maximum is taken over all (possibly unbounded) distinguishers. In case the maximum is taken over all PPT distinghuishers, we sometimes speak of computational distance. For two ensembles $\mathcal{X} = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$, we write $\mathcal{X} \equiv \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are identically distributed, $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are statistically close (i.e., their statistical distance is bounded by a negligible function of the security parameter), and $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable.

The following lemma follows directly from the definition of statistical distance.

Lemma 1. *Let A and B be a pair of random variables, and E be an event defined over the probability space of A and B. Then, $\Delta(A, B) \leqslant \Delta(A, B|E) + \Pr[\neg E]$.*

2.2 Non-Interactive Zero-Knowledge

We recall the notion of a non-interactive zero-knowledge proof of knowledge (NIZK-PoK) system for an *NP* relation R (with corresponding language L). A NIZK-PoK is a tuple $\mathcal{NIZK} := (\ell, \mathsf{Prove}, \mathsf{Ver})$ specified as follows: (i) At setup, a random common reference string (CRS)

$\omega \leftarrow\$ \{0,1\}^{\ell(\kappa)}$ is generated where ℓ is polynomial in the security parameter; (ii) Algorithm Prove takes as input the CRS together with some pair $(x, w) \in R_L$, and returns a proof $\pi \leftarrow\$ \mathsf{Prove}(\omega, x, w)$; (iii) Algorithm Ver takes as input the CRS together with some pair (x, π), and returns a decision bit $\mathsf{Ver}(\omega, x, \pi)$.

The definition below is adapted from [RS91, SP92].

Definition 1. We say that $\mathcal{NIZK} = (\ell, \mathsf{Prove}, \mathsf{Ver})$ is a NIZK-PoK system for the relation $R \subseteq NP$, if the following conditions are met.

(a) *Perfect Completeness.* For all pairs $(x, w) \in R$, we have that $\mathsf{Ver}(\omega, x, \mathsf{Prove}(\omega, x, w)) = 1$ with probability 1 over the coin tosses of the prover algorithm and the choice of the CRS.

(b) *Unbounded Zero-Knowledge.* There exists a simulator $\mathcal{Z} = (\mathcal{Z}_0, \mathcal{Z}_1)$ such that for all PPT distinguishers $\mathcal{D}$, and all auxiliary strings $z \in \{0,1\}^*$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:

$$\Big| \Pr\left[\mathcal{D}^{\mathsf{Prove}(\omega,\cdot,\cdot)}(\omega, z; r) = 1 :\ \omega \leftarrow\$ \{0,1\}^{\ell}(\kappa)\right] \\ - \Pr\left[\mathcal{D}^{\mathsf{Simu}(\cdot,\cdot,\vartheta,z)}(\omega, z; r) = 1 :\ (\omega, \vartheta) \leftarrow\$ \mathcal{Z}_0(1^\kappa)\right] \Big| \leqslant \mathit{negl}(\kappa),$$

where $\mathsf{Simu}(x, w, \vartheta, z) := \mathcal{Z}_2(\vartheta, x, z)$.

(c) *Knowledge Soundness.* There exists a PPT extractor $\mathcal{K} = (\mathcal{K}_0, \mathcal{K}_1)$ such that the distribution induced by $\mathcal{K}_0(1^\kappa)$ is negligibly close (in statistical distance) to the uniform distribution over $\{0,1\}^{\ell(\kappa)}$. Moreover, for all PPT provers $\mathcal{P}^*$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that

$$\Pr[(x, w) \in R :\ (\omega, \vartheta) \leftarrow\$ \mathcal{K}_0(1^\kappa); (x, \pi) \leftarrow\$ \mathcal{P}^*(\omega); w \leftarrow\$ \mathcal{K}_1(\omega, \vartheta, x, \pi)] \\ \geq \Pr[\mathsf{Ver}(\omega, x, \pi) = 1 :\ \omega \leftarrow\$ \{0,1\}^{\ell(\kappa)}, (x, \pi) \leftarrow\$ \mathcal{P}^*(\omega)] - \nu(\kappa).$$

2.3 Commitment Schemes

A (non-interactive) commitment scheme is a PPT algorithm Com that upon input the security parameter, a message m (within a space of possible messages), and randomness $r \leftarrow\$ \{0,1\}^*$ produces a commitment com. For a formal definition of commitment scheme we refer the reader to the text book of Goldereich [Gol01]. We require Com to satisfy the following properties.

Perfect Binding. For any (possibly unbounded) adversary $\mathcal{A}$, the following holds:

$$\Pr[\mathsf{Com}(1^\kappa, m; r) = \mathsf{Com}(1^\kappa, m'; r') \wedge m \neq m' :\ (m, r, m', r') \leftarrow\$ \mathcal{A}] = 0.$$

Computational Hiding. For any two messages m, m' in the message space the ensembles $\{\mathsf{Com}(1^\kappa, m)\}_{\kappa \in \mathbb{N}}$ and $\{\mathsf{Com}(1^\kappa, m')\}_{\kappa \in \mathbb{N}}$ are computationally indistinguishable.

2.3.1 Interactive Protocols

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be an NP-relation, naturally defining a language $L_R := \{x : \exists w \text{ s.t. } (x, w) \in R\}$. We are typically interested in efficiently samplable relations, for which there exists a PPT algorithm SamR taking as input the security parameter (and random coins r) and outputting a pair $(x, w) \in R$. An interactive protocol $\Pi = (\mathcal{P}, \mathcal{V})$ for R features a prover $\mathcal{P}$ (holding a value $x \in L_R$ together with a corresponding witness w) and a verifier $\mathcal{V}$ (holding x), where the goal of the prover is to convince the verifier that $x \in L_R$. At the end of the protocol execution, the verifier outputs either `acc` or `rej`. We write $\langle \mathcal{P}(1^\kappa, x, w), \mathcal{V}(1^\kappa, x) \rangle$ for the random variable corresponding to the verifier's verdict, and $\mathcal{P}(1^\kappa, x, w) \leftrightarrows \mathcal{V}(1^\kappa, x)$ for the random variable corresponding to a transcript of protocol Π on input (x, w).

Unless stated otherwise, all interactive protocols considered in this paper are *secret-coin*, meaning that the verifier's strategy depends on a secretly kept random tape. We also call Π a ρ-round protocol if the protocol consists of ρ rounds, where each round features a message from the verifier to the prover and viceversa.

3 Predictable Arguments of Knowledge

We start by defining Predictable Arguments of Knowledge (PAoK) in Section 3.1as multi-round interactive protocols in which the verifier generates a challenge (to be sent to the prover) and can at the same time predict the prover's answer to that challenge; we insist on (computational) extractable security, meaning that from any prover convincing a verifier with some probability we can extract a witness with probability related to the prover's success probability.

The main result of this section is that PAoK can be assumed without loss of generality to be extremely laconic (i.e., the prover sends a single bit and the protocol consists of a single round of communication). More in detail, in Section 3.2, we show that any multi-round PAoK can be squeezed into a one-round PAoK. In Section 3.3 we show that, for any $\ell \in \mathbb{N}$, the existence of a PAoK where the prover answer is of length ℓ bits implies the existence of a laconic PAoK.

3.1 The Definition

In a multi-round protocol the verifier produces many challenges $\mathbf{c} = (c_1, \dots, c_\rho)$. W.l.o.g. in a predictable argument, we can assume that all the challenges are generated together and then forwarded one-by-one to the prover; this is because the answers are known *in advance*. Specifically, a ρ-round predictable argument is fully specified by a tuple of algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp})$, as described below:

1. $\mathcal{V}$ samples $(\mathbf{c}, \mathbf{b}) \leftarrow\!\!\$\; \mathsf{Chall}(1^\kappa, x)$, where $\mathbf{c} := (c_1, \dots, c_\rho)$ and $\mathbf{b} := (b_1, \dots, b_\rho)$.
2. For all $i \in [\rho]$ in increasing sequence:
   - $\mathcal{V}$ forwards c_i to $\mathcal{P}$;
   - $\mathcal{P}$ computes $(a_1, \dots, a_i) := \mathsf{Resp}(1^\kappa, x, w, c_1, \dots, c_i)$ and forwards a_i to $\mathcal{V}$;
   - $\mathcal{V}$ checks that $a_i = b_i$, and returns `rej` if this is not the case.
3. If all challenges are answered correctly, $\mathcal{V}$ returns `acc`.

Notice that the algorithm Resp takes as input all challenges up-to round i in order to generate the i-th answer.[11]

We say that prover $\mathcal{P}$ and verifier $\mathcal{V}$, running the protocol above, *execute a PAoK* Π upon input security parameter 1^κ, common input x, and prover's private input w; we denote with $\langle \mathcal{P}(1^\kappa, x, w), \mathcal{V}(1^\kappa, x)\rangle_\Pi$ (or, when Π is clear from the context, simply $\langle \mathcal{P}(1^\kappa, x, w), \mathcal{V}(1^\kappa, x)\rangle$) the output of such interaction. We say that a prover $\mathcal{P}$ *succeeds* on the instance x and auxiliary input w if $\langle \mathcal{P}(1^\kappa, x, w), \mathcal{V}(1^\kappa, x)\rangle = \mathtt{acc}$. We give a granular definition of extractability that is parametrized by an efficient instance sampler $\mathcal{S}$, and that roughly says that the protocol is sound and moreover sampled instances are extractable. Here, the sampler is simply an algorithm taking as input the security parameter and auxiliary input $z_S \in \{0,1\}^*$, and outputting an instance x together with auxiliary information $aux \in \{0,1\}^*$.

Definition 2 (Predictable Arguments of Knowledge)**.** Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be a ρ-round predictable argument for an *NP* relation R, with ℓ-bit prover's answer. Consider the properties below.

[11] In the description above we let Resp output also all previous answers $a_1, \dots, a_{i-1}$; while this is not necessary it can be assumed w.l.o.g. and will simplify the proof of Theorem 1.

Completeness: There exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that for all sequences $\{(x_\kappa, w_\kappa)\}_{\kappa \geqslant 0}$ where $(x_\kappa, w_\kappa) \in R$, we have that:

$$\Pr_{\mathcal{P},\mathcal{V}}\left[\langle \mathcal{P}(1^\kappa, x_\kappa, w_\kappa), \mathcal{V}(1^\kappa, x_\kappa)\rangle = \mathtt{rej}\right] \leqslant \nu(\kappa).$$

$(\mathcal{S}, f, \epsilon)$-Knowledge soundness: For all PPT provers $\mathcal{P}^*$ there exists a PPT extractor $\mathcal{K}$ such that for all auxiliary inputs $z_P, z_S \in \{0,1\}^*$ the following holds. Whenever

$$p(\kappa) := \Pr_{\mathcal{P}^*,\mathcal{V},r_S}\left[\langle \mathcal{P}^*(1^\kappa, aux, x, z_P), \mathcal{V}(x)\rangle = \mathtt{acc} : (x, aux) := \mathcal{S}(1^\kappa, z_S; r_S)\right] > \epsilon(\kappa)$$

then

$$\Pr_{\mathcal{K},r_S}\left[\begin{array}{c} \exists w \text{ s.t. } f(w) = y \\ (x,w) \in R \end{array} : \begin{array}{c} (x, aux) := \mathcal{S}(1^\kappa, z_S; r_S), \\ y \leftarrow\!\!\$\, \mathcal{K}(1^\kappa, x, z_P, z_S, aux) \end{array}\right] \geqslant p(\kappa) - \epsilon(\kappa).$$

We call Π a ρ-round $\mathcal{S}$-PAoK for R, if Π satisfies completeness and $(\mathcal{S}, f, \epsilon)$-knowledge soundness for any efficient computable function f, and moreover $\epsilon - 2^{-\rho\ell}$ is negligible. We call Π an $\mathcal{S}$-PAoK for R, if Π is a 1-round $\mathcal{S}$-PAoK and we call it a *laconic* $\mathcal{S}$-PAoK if Π is an $\mathcal{S}$-PAoK and $\ell = 1$. Sometimes we also say that Π is a ρ-round $(f, \mathcal{S})$-PAoK if knowledge soundness holds for a specific function f.

Consider the dummy sampler $\mathcal{S}_{\mathsf{dummy}}$ that parses its input z_S as (x, aux) and then outputs the pair (x, aux). We call Π a ρ-round (f, ϵ)-PAoK for R, if Π satisfies completeness and $(\mathcal{S}_{\mathsf{dummy}}, f, \epsilon)$-knowledge soundness. We say that Π is a ρ-round PAoK for R, if Π is a ρ-round $\mathcal{S}_{\mathsf{dummy}}$-PAoK for R.

The reason why the above definition is parametrized by the function f instead of considering the relation $R' = \{(x, y) : \exists w \text{ s.t. } (x, w) \in R \wedge y = f(w)\}$ is that such a relation might not be an NP-relation (as it might be hard to check whether $\exists w$ s.t. $(x, w) \in R \wedge y = f(w)$. Our definition, instead, ensures that the honest prover knows w but we can only extract $f(w)$. Also note that, in the above definition, the prover $\mathcal{P}^*$ takes as input the auxiliary information returned by the sampler.

3.2 On Multi-Round PAoK

In this section we show that multi-round PAoK can be squeezed into a one-round PAoK (maintaining knowledge soundness).

Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be a ρ-round PAoK. Consider the following protocol between prover $\tilde{\mathcal{P}}_n$ and verifier $\tilde{\mathcal{V}}_n$—let us call it the *collapsed protocol* for future reference—for a parameter $n \in \mathbb{N}$ to be determined later:

- Repeat the following sub-protocol $\tilde{\Pi} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$ in parallel for all $j \in [n]$:
  - $\tilde{\mathcal{V}}$ runs $(\mathbf{c}^j, \mathbf{b}^j) \leftarrow\!\!\$\, \mathsf{Chall}(1^\kappa, x)$; let $\mathbf{c}^j = (c_1^j, \ldots, c_\rho^j)$ and similarly $\mathbf{b}^j = (b_1^j, \ldots, b_\rho^j)$. Then, $\tilde{\mathcal{V}}$ samples a random index $i_j^* \leftarrow\!\!\$\, [\rho]$, and forwards $(c_1^j, \ldots, c_{i_j^*}^j)$ to $\tilde{\mathcal{P}}$.
  - $\tilde{\mathcal{P}}$, given a pair (x, w) and challenges $(c_1^j, \ldots, c_{i_j^*}^j)$, computes $(a_1^j, \ldots, a_{i_j^*}^j) \leftarrow\!\!\$\, \mathsf{Resp}(1^\kappa, x, w, c_1^j, \ldots, c_{i_j^*}^j)$ and forwards $(a_1^j, \ldots, a_{i_j^*}^j)$ to $\tilde{\mathcal{V}}$.
  - $\tilde{\mathcal{V}}$ is said to accept the j-th parallel execution if and only if $a_i^j = b_i^j$ for all $i \in [i_j^*]$
- Return $\mathtt{acc}$ if and only if all parallel executions are accepting.

We write $\tilde{\Pi}_n := (\tilde{\mathcal{P}}_n, \tilde{\mathcal{V}}_n)$ for the n-fold repetition of the sub-protocol $\tilde{\Pi} = (\tilde{\mathcal{P}}, \tilde{\mathcal{V}})$. Note that the sub-protocol $\tilde{\Pi}$ is the one-round protocol (described above) that simply cuts the multi-round protocol Π to a random round. We show the following theorem :

Theorem 1. *For any polynomial $\rho(\cdot)$ and any function f if Π is a $\rho(\kappa)$-round f-PAoK, then the above defined collapsed protocol $\tilde{\Pi}_n = (\tilde{\mathcal{P}}_n, \tilde{\mathcal{V}}_n)$ with parameter $n = \omega(\rho \log \kappa)$ is an f-PAoK.*

Proof overview. For simplicity, assume that Π is a $\frac{1}{3}$-PAoK for the relation R. We claim that the knowledge error of the collapsed protocol is not bigger than $1 - \frac{2}{3\rho}$. To see this, consider a prover $\mathcal{P}^*$ for the original protocol Π which at the i-th iteration (where $i \in [\rho]$) forwards the the challenge $c_1, \dots, c_i$ to a malicious prover $\tilde{\mathcal{P}}^*$ for the collapsed protocol. Notice that conditioned on $i^* = i$ the challenge has exactly the same distribution as a challenge for the collapsed protocol. The prover $\mathcal{P}^*$ fails if the malicious prover $\tilde{\mathcal{P}}^*$ of the collapsed protocol answered wrongly at least one of the queries that he received. So if we suppose that $\tilde{\mathcal{P}}^*$ succeeds with probability strictly bigger than $1 - \frac{2}{3\rho}$, then, by the union bound, the failing probability of $\mathcal{P}^*$ is strictly bounded by $\frac{2}{3\rho} \cdot \rho$, therefore $\mathcal{P}^*$ succeeds with probability strictly bigger than $\frac{1}{3}$.

Finally, we can make the knowledge soundness error of the collapsed protocol negligible via parallel repetition. It is important to notice that parallel repetition, in general, does not amplify soundness for argument systems [BIN97, PW12]. Luckily, it does so (at an exponential rate) in the special case of secret-coin one-round arguments (such as PAoK) [BIN97].

Formal proof. More precisely, theproof of the above theorem relies on the well-known fact that parallel repetition decreases the (knowledge) soundness error of one-round arguments at an exponential rate.

Lemma 2 (Theorem 4.1 of [BIN97], adapted to one-round protocols)**.** *Let $\Pi = (\mathcal{P}, \mathcal{V})$ be a one-round argument of knowledge and denote by $\Pi_n = (\mathcal{P}_n, \mathcal{V}_n)$ the one-round protocol that consists of the n-fold repetition of the initial protocol Π. Suppose $0 < \alpha, \beta < 1$ and $n \geqslant 2$ is an integer. Suppose $\alpha > (16/\beta) \cdot e^{-\beta \cdot n/128}$. Then there is an oracle algorithm $\mathcal{R}$ such that for any prover $\mathcal{P}^*$, verifier $\mathcal{V}$ and input string x, the following is true: If $\Pr[\langle \mathcal{P}^*(1^\kappa, x, aux), \mathcal{V}_n(x) \rangle = \mathsf{acc}] \geqslant 2\alpha$ then $\Pr[\langle \mathcal{R}^{\mathcal{P}^*}(1^\kappa, x, aux), \mathcal{V}(x) \rangle = \mathsf{acc}] \geqslant 1 - \beta$. Furthermore, $\mathcal{R}^{\mathcal{P}^*}$ runs in time poly$(n, |x|, \alpha^{-1})$.*

Proof of Theorem 1. Let $\tilde{\mathcal{P}}_n^*$ be a prover for the collapsed protocol such that for some x and z succeeds with probability at least κ^{-c} for some constant c. Let $\alpha = \frac{1}{2}\kappa^{-c}$ and $\beta = \frac{1}{2\rho}$, notice that setting $n = \omega(\rho \log \kappa)$ the following equation holds for κ big enough:

$$\tfrac{1}{2}\kappa^{-c} = \alpha > (16/\beta) \cdot e^{-\beta \cdot n/128} = 32\rho \cdot e^{-\omega(\log \kappa)/256}.$$

We can apply Lemma 2 with the parameters α and β set as above. Therefore, consider a single instance of the sub-protocol $\tilde{\Pi}$, the prover $\mathcal{R}^{\tilde{\mathcal{P}}_n^*}$ succeeds with probability $1 - \beta = 1 - \frac{1}{2\rho}$.

We build a prover $\mathcal{P}^*$ for Π that succeeds with probability $\frac{1}{2}$. Specifically, Let $\tilde{\mathcal{P}}^* := \mathcal{R}^{\tilde{\mathcal{P}}_n^*}$ and let $\mathcal{P}^*$ interact with the verifier $\mathcal{V}$ of the multi-round protocol as follow:

1. $\mathcal{V}$ samples $(\mathbf{c}, \mathbf{b}) \leftarrow\$ \mathsf{Chall}(1^\kappa, x)$, where $\mathbf{c} := (c_1, \dots, c_\rho)$ and $\mathbf{b} := (b_1, \dots, b_\rho)$.
2. For all $i \in [\rho]$ in increasing sequence:
    - Upon input challenge c_i from the verifier $\mathcal{V}$, prover $\mathcal{P}^*$ runs internally $\tilde{\mathcal{P}}^*$ on input $(1^\kappa, x)$ and challenge $(c_1, \dots, c_i)$. If $\tilde{\mathcal{P}}^*$ outputs $(a_1, \dots, a_i)$, then $\mathcal{P}^*$ forwards a_i to $\mathcal{V}$; otherwise it aborts.

Rewriting explicitly the acceptance probability of $\tilde{\mathcal{P}}^*$ in the collapsed protocol on (x, z):

$$\Pr\left[\tilde{\mathcal{P}}^*(1^\kappa, x, z, c_1, \dots, c_i) = (b_1, \dots, b_i) : (\mathbf{c}, \mathbf{b}) \leftarrow\$ \mathsf{Chall}(1^\kappa, x), i \leftarrow\$ [\rho]\right] \geqslant 1 - \tfrac{1}{2\rho}.$$

Let W_i be the event that $a_i = b_i$ in the interaction between $\mathcal{P}^*$ and $\mathcal{V}$ described above. We can write:

$$\begin{aligned}
&\Pr[\langle \mathcal{P}^*(1^\kappa, x, z), \mathcal{V}(1^\kappa, x)\rangle = \mathsf{acc}] && (1)\\
&= \Pr[\forall i \in [\rho] : W_i] = 1 - \Pr[\exists i \in [\rho] : \neg W_i] \geqslant 1 - \sum_{i \in [\rho]} \Pr[\neg W_i]\\
&= 1 - \rho \cdot \mathbb{E}_{i \leftarrow\!\!\$\, [\rho]}\left[\Pr[\mathcal{P}^*(1^\kappa, x, c_1, \ldots, c_i) \neq a_i : (\mathbf{c}, \mathbf{b}) \leftarrow\!\!\$\, \mathsf{Chall}(1^\kappa, x)]\right]\\
&\geqslant 1 - \left(\tfrac{1}{2\rho}\right) \cdot \rho = \tfrac{1}{2}.
\end{aligned}$$

where the equations above follow by the definition of average and by our assumption on the success probability of $\tilde{\mathcal{P}}^*$ on (x, z). Notice that for any successful $\tilde{\mathcal{P}}^*_n$ we can define an extractor that is the same extractor for the machine $\mathcal{P}^*$ executing $\tilde{\mathcal{P}}^* = \mathcal{R}^{\tilde{\mathcal{P}}^*_n}$ as a subroutine. Moreover, since $\tilde{\mathcal{P}}^*_n$ succeeds with probability κ^{-c} then $\mathcal{P}^*$ runs in polynomial time. □

3.3 Laconic PAoK

We show that laconic PAoK (where the size of the prover's answer is $\ell = 1$ bit) are in fact equivalent to PAoK.

Theorem 2. *Let R be an NP relation. If there exists a PAoK for R then there exists a* laconic *PAoK for R.*

The proof of the theoremrelies on the Goldreich-Levin Theorem [Gol01, Theorem 2.5.2]. Here is the intuition. Let $(\mathcal{P}, \mathcal{V})$ be a PAoK with ℓ-bit answers. We define a new PAoK $(\mathcal{P}', \mathcal{V}')$ where the verifier $\mathcal{V}'$ runs $\mathcal{V}$ in order to obtain a pair (c, b), samples randomness r, and defines the new predicted answer to be the inner product between b and r. Given challenge (c, r) the prover $\mathcal{P}'$ simply runs $\mathcal{P}$ in order to obtain a and defines the answer to be the inner product between a and r. Knowledge soundness follows by the Goldreich-Levin theorem. In particular, we use the fact that Goldreich-Levin theorem holds not only for injective one-way functions, but more generally for any relation, provided that for any instance there exists only one witness.

Lemma 3 (Goldreich-Levin Theorem)**.** *Consider a relation $R_{\kappa,\ell} \subseteq \{(c, b) : c \in \{0,1\}^\kappa, b \in \{0,1\}^\ell\}$ with corresponding language $L_{\kappa,\ell}$, such that for any instance $c \in L_{\kappa,\ell}$ there exists only one valid witness b for c. There exists a PPT inverter $\mathcal{I}$ and a non-zero polynomial $q(\cdot)$ such that, for any machine $\mathcal{P}$ and any $c \in \{0,1\}^\kappa$, whenever $p(c) := \Pr\left[\mathcal{P}(c, r) = \langle b, r\rangle : (c, b) \in \mathcal{R}_{\kappa,\ell}; r \leftarrow\!\!\$\, \{0,1\}^\ell\right]$ (where $\langle \cdot, \cdot \rangle$ denotes the inner product over the binary field) then $\Pr[\mathcal{I}^{\mathcal{P}(c,\cdot)}(1^\ell, c) = b \ \wedge \ (c, b) \in R_{\kappa,\ell}] \geqslant q\left(p(c) - \frac{1}{2}\right)$.*

Proof of Theorem 2. Consider $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ to be a PAoK for R, with ℓ-bit prover's answer for some $\ell = poly(\kappa)$. In what follows when we write $\langle a, b\rangle$ for strings $a, b \in \{0,1\}^\ell$ we mean the inner product between a and b when interpreted as vectors in the binary field. Define the protocol $\Pi' = (\mathsf{Chall}', \mathsf{Resp}')$ described below:

- Upon input $(1^\kappa, x)$, let $\mathsf{Chall}'(1^\kappa, x) := (c', b')$ where $c' = (c, r)$ and $b' = \langle b, r\rangle$ for a random $r \leftarrow\!\!\$\, \{0,1\}^\ell$.
- Upon input $(1^\kappa, x, w, c')$, let $\mathsf{Resp}'(1^\kappa, x, w, c') := \langle a, r\rangle$ where $c' = (c, r)$ and $a = \mathsf{Resp}(1^\kappa, x, w, c)$.

Clearly, Π' is laconic. Consider now the relation $R^* = \{(c, b) : \exists r \text{ s.t. } (c, b) = \mathsf{Chall}(1^\kappa, x; r)\}$. Given a prover $\mathcal{P}'$ for Π' we can define the prover $\mathcal{P}^*$ that upon input the instance x and a challenge c runs the inverter $\mathcal{I}(1^\ell, c)$ from Lemma 3 and forwards its oracle queries to $\mathcal{P}'(1^\kappa, x, z, \cdot)$.

By Lemma 3 we have that such a prover runs in polynomial time if $\mathcal{P}'$ does, and for every challenge c its success probability is polynomially related to the success probability of $\mathcal{P}'$. Therefore, if $\mathcal{P}'$ succeeds with noticeable probability so does $\mathcal{P}^*$. The statement follows. □

4 Constructing PAoK

We explore constructions of PAoK. In Section 4.1 we investigate the relationship between PAoK and Extractable Witness Encryption [GGSW13, GKP$^+$13]. In particular, we establish the equivalence between the two notions.

In Section 4.2 we show that we can construct a PAoK from any extractable hash-proof system [Wee10] (Ext-HPS); if the Ext-HPS is defined w.r.t. a relation R, we obtain a PAoK for a related relation R' where R and R' share the same x, and the witness for x w.r.t. R' is the randomness used to sample the instance $(x, w) \in R$.

In Section 4.3, we focus on constructing PAoK for so-called random self-reducible relations. In particular, we show that, for such relations, a fully-extractable PAoK can be obtained by generically leveraging a PAoK for a (much weaker) specific sampler (which depends on the random self-reducible relation).

Finally, in Section 4.4, we show that a PAoK for a specific sampler can be obtained generically by using a differing-input obfuscator [BST14] for a related (specific) circuit sampler.

4.1 Equivalence to Extractable Witness Encryption

We show that full-fledged PAoK imply extractable witness encryption (Ext-WE), and viceversa. We start by recalling the definition of Ext-WE, taken from [GGHW14][12].

Extractable Witness Encryption. Let R be an NP-relation. A WE scheme $\Pi = (\mathsf{Encrypt}, \mathsf{Decrypt})$ for R (with message space $\mathcal{M} = \{0,1\}$) consists of two PPT algorithms, specified as follows:[13] (i) Algorithm $\mathsf{Encrypt}$ takes as input a security parameter 1^κ, a value $x \in \{0,1\}^*$, and a message $\beta \in \{0,1\}$, and outputs a ciphertext γ; (ii) Algorithm $\mathsf{Decrypt}$ takes as input a security parameter 1^κ, a ciphertext γ, a value $w \in \{0,1\}^*$, and outputs a message $\beta \in \{0,1\}$ or a special symbol $\perp$.

Definition 3 (Ext-WE). Let R be an NP-relation, and $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ be a WE scheme for R. We say that Π_{WE} is an Ext-WE scheme for R if the following requirements are met.

Correctness: For any $(x, w) \in R_L$ and $\beta \in \{0,1\}$, we have that $\mathsf{Decrypt}(1^\kappa, w, \mathsf{Encrypt}(1^\kappa, x, \beta)) = \beta$ with probability one.

Extractable Security: For any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and for any noticeable function $\epsilon(\cdot)$, there exists a non-uniform extractor $\mathcal{K}$ and a non-zero polynomial $q(\cdot)$ such that the following holds. For any auxiliary information $z \in \{0,1\}^*$ and for any tuple $(x, st) \leftarrow\!\$\ \mathcal{A}_0(1^\kappa, z)$, whenever

$$\Pr\left[\mathcal{A}_1(1^\kappa, st, x, \mathsf{Encrypt}(1^\kappa, x, \beta), z) = \beta :\ \beta \leftarrow\!\$\ \{0,1\}\right] \geqslant \frac{1}{2} + \epsilon(\kappa)$$

we have $\Pr[(x, \mathcal{K}(1^\kappa, x, z)) \in R_L] \geqslant \epsilon(\kappa)$.

Theorem 3. *Let R be an NP-relation. There exists a PAoK for R if and only if there exists an Ext-WE scheme for R.*

Intuitively, the equivalence between PAoK and Ext-WE can be established as follows:

- From Ext-WE to PAoK we encrypt a random bit a using the encryption scheme and then ask the prover to return a.

[12] More precisely, we make the slightly stronger assumption that the value of the extraction probability is at least the advantage of the adversary.

[13] WE for arbitrary-length messages can be obtained encrypting each bit of the plaintext independently.

- From PAoK to Ext-WE, we first make the PAoK extremely laconic, then we generate a challenge/answer pair (c, a) for the PAoK, and encrypt a single bit β as $(c, a \oplus \beta)$.

Proof of Theorem 3. Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be a PAoK for the relation R. Without loss of generality, by our analysis in Section 3, we can assume that the PAoK is laconic (i.e., the output of Resp is a single bit $a \in \{0,1\}$). Consider the following construction of an Ext-WE scheme $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ for R (with message space $\mathcal{M} = \{0,1\}$):

- Upon input $1^\kappa, x$ and message β, define $\mathsf{Encrypt}(1^\kappa, x, \beta) := (c, \beta \oplus b) := \gamma$ where $(c, b) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa, x)$.
- Upon input $1^\kappa, w, \gamma$, where $\gamma = (\gamma_1, \gamma_2)$, define $\mathsf{Decrypt}(1^\kappa, w, \gamma) = \gamma_2 \oplus a$ where $a \leftarrow\!\!\$\ \mathsf{Resp}(1^\kappa, x, w, \gamma_1)$.

Let $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ be an adversary for the WE scheme. Assume that there exists a noticeable function $\epsilon(\cdot)$ such that

$$\Pr\left[\mathcal{A}_1(1^\kappa, st, x, \gamma, z) = \beta :\ \beta \leftarrow\!\!\$\ \{0,1\}; \gamma \leftarrow\!\!\$\ \mathsf{Encrypt}(1^\kappa, x, \beta)\right] \geqslant \frac{1}{2} + \epsilon(\kappa)$$

for $(st, x) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa, z)$ (where $z \in \{0,1\}^*$ is the auxiliary input). We use $\mathcal{A}$ to construct a prover $\mathcal{P}^*$ attacking knowledge soundness of Π. Prover $\mathcal{P}^*$ first runs $(st, x) \leftarrow\!\!\$\ \mathcal{A}_0(1^\kappa, z)$, and then interacts with the honest verifier of Π on common input x, as follows:

1. Receive challenge c from the verifier.
2. Sample $\beta' \leftarrow\!\!\$\ \{0,1\}$ and run $\mathcal{A}_1(1^\kappa, st, x, \gamma, z)$ on $\gamma := (c, \beta')$, obtaining a bit β.
3. Send $a := \beta \oplus \beta'$ to the verifier.

For the analysis, note that the ciphertext simulated by $\mathcal{P}^*$ has the right distribution (in particular, the second component is a random bit). Since $\beta' = \beta \oplus b$ we get that $\mathcal{P}^*$ outputs $a = b$ with probability at least $1/2 + \epsilon(\kappa)$ and thus $\mathcal{P}^*$ convinces $\mathcal{V}$ with probability $p(\kappa) \geq 1/2 + \epsilon(\kappa)$. We are now in a position to run the extractor $\mathcal{K}$ of Π, and hence we obtain a valid witness $w \leftarrow\!\!\$\ \mathcal{K}(1^\kappa, x, z)$ with probability $\epsilon(\kappa)$. The statement follows.

Conversely, let $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ be an Ext-WE scheme for the relation R, with message space $\mathcal{M} = \{0,1\}$. Consider the following construction of a PAoK $\Pi = (\mathsf{Chall}, \mathsf{Resp})$:

- Upon input $1^\kappa, x$, define $\mathsf{Chall}(1^\kappa, x) := (\mathsf{Encrypt}(1^\kappa, x, b), b)$ where $b \leftarrow\!\!\$\ \{0,1\}$.
- Upon input $1^\kappa, x, w, c$, define $\mathsf{Resp}(1^\kappa, x, w, c) := \mathsf{Decrypt}(1^\kappa, w, c)$.

Fix any x and let $\mathcal{P}^*$ be a malicious prover for the PAoK. Assume that there exists a polynomial $p(\cdot)$ such that

$$p(\kappa) := \Pr\left[\langle \mathcal{P}^*(1^\kappa, x, z), \mathcal{V}(1^\kappa, x)\rangle = \mathsf{acc}\right] \geq \epsilon(\kappa).$$

where $z \in \{0,1\}^*$ is the auxiliary input. We use $\mathcal{P}^*$ to construct an adversary $\mathcal{A} := (\mathcal{A}_0, \mathcal{A}_1)$ attacking extractable security of Π_{WE}. Adversary $\mathcal{A}_0(1^\kappa, z)$ outputs x, and then $\mathcal{A}_1$ is given a challenge ciphertext γ that is either an encryption of $\beta = 0$ or an encryption of $\beta = 1$ (under x), and its goal is to guess β. To do so $\mathcal{A}$ proceeds as follows:

1. Forward γ to $\mathcal{P}^*$.
2. Let a be the answer sent by $\mathcal{P}^*$; output $\beta := a$.

For the analysis, note that the challenge simulated by $\mathcal{A}_1$ has the right distribution (in particular, it is a witness encryption of a random bit). Since $a = b = \mathsf{Decrypt}(1^\kappa, x, w, \gamma)$ with probability at least $p(\kappa)$, we get that $\mathcal{A}_1$ guesses β with at least the same probability. We are now in a position to run the extractor $\mathcal{K}$ of Π_{WE}, and hence we obtain a valid witness $w \leftarrow\!\!\$\ \mathcal{K}(1^\kappa, x, z)$ with probability $p(\kappa) - \epsilon(\kappa)$. The statement follows. A similar argument shows that Π is a weak PAoK whenever Π_{WE} is a weak Ext-WE. □

4.2 Construction from Extractable Hash-Proof Systems

The definition below is adapted from [Wee10].

Definition 4 (Ext-HPS)**.** Let $\mathcal{H} = \{h_{pk}\}$ be a set of hash functions indexed by a public key pk, and let R be an *NP*-relation. An extractable hash-proof system for R is a tuple of PPT algorithms $\Pi_{\mathsf{HPS}} := (\mathsf{SetupHash}, \mathsf{SetupExt}, \mathsf{Ext}, \mathsf{Pub}, \mathsf{Priv})$ such that the following properties are satisfied.

Public evaluation: For all $(pk, sk) \leftarrow \mathsf{SetupExt}(1^\kappa)$, and $(x, w) \leftarrow \mathsf{SamR}(1^\kappa; r)$, we have $\mathsf{Pub}(1^\kappa, pk, r) = h_{pk}(x)$.

Extraction mode: For all $(pk, sk) \leftarrow \mathsf{SetupExt}(1^\kappa)$ and all (x, π), we have that $\pi = h_{pk}(x) \Leftrightarrow (x, \mathsf{Ext}(1^\kappa, sk, x, \pi)) \in R$.

Hashing mode: For all $(pk, sk) \leftarrow \mathsf{SetupHash}(1^\kappa)$, and for all $(x, w) \in R$, we have that $\mathsf{Priv}(1^\kappa, sk, x) = h_{pk}(x)$.

Indistinguishability: The ensembles $\{pk : (pk, sk) \leftarrow \mathsf{SetupHash}(1^\kappa)\}_{\kappa \in \mathbb{N}}$ and $\{pk : (pk, sk) \leftarrow \mathsf{SetupExt}(1^\kappa)\}_{\kappa \in \mathbb{N}}$ are statistically indistinguishable.

Let R be an efficiently samplable relation with sampling algorithm SamR. Define the relation R', such that $(x, w') \in R'$ if and only if $(x, w) \in R$ where $(x, w) := \mathsf{SamR}(1^\kappa; w')$. Consider the following pair of PPT algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp})$, defining a one-round predictable argument for R' (as described in Section 3.1).

1. Algorithm $\mathsf{Chall}(1^\kappa, x)$ runs $(pk, sk) \leftarrow \mathsf{SetupHash}(1^\kappa)$, and defines $c := pk$ and $b := \mathsf{Priv}(1^\kappa, sk, x)$.
2. Algorithm $\mathsf{Resp}(1^\kappa, x, w', c)$ defines $a := \mathsf{Pub}(1^\kappa, pk, w')$.

Theorem 4. *Let R, R' and* SamR *be as above. Assume that* Π_{HPS} *is an Ext-HPS for the relation R. Then* $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ *as defined above is an f-PAoK for the relation R' and where $f(\cdot)$ returns the second output of* $\mathsf{SamR}(\cdot)$.

Proof. Completeness follows by the correctness property of the hashing mode of the underlying HPS. In order to show knowledge soundness, we consider a mental experiment where algorithm Chall is defined differently. In particular, the verifier samples $(pk, sk) \leftarrow \mathsf{SetupExt}(1^\kappa)$ using the extraction mode instead of the hashing mode. By the indistinguishability property of the HPS this results in a statistically close distribution.

Now, we can define the extractor $\mathcal{K}$ of the PAoK as follows. Let $\mathcal{P}^*$ be a PPT algorithm such that $\langle \mathcal{P}^*(1^\kappa, x, z), \mathcal{V}(1^\kappa, x)\rangle_\Pi = \mathtt{acc}$ with probability $p(\kappa)$, where $\mathcal{P}^*$ uses auxiliary input $z \in \{0,1\}^*$. Define $\mathcal{K}(1^\kappa, x, z) := \mathsf{Ext}(1^\kappa, sk, x, a)$ where a is the message sent by $\mathcal{P}^*$. By definition of protocol Π we get that whenever $\mathcal{P}^*$ succeeds then $c = h_{pk}(x) = a$. Thus the extraction property of the HPS implies that $w \leftarrow\!\!\$\, \mathsf{Ext}(1^\kappa, sk, x, a)$ is a valid witness for x, i.e. $R(x, w) = 1$ with probability 1. The proof now follows by the fact that for all w' such that $\mathsf{SamR}(1^\kappa; w') = (x, w)$, we also have $R'(x, w') = 1$. □

Instantiations. We consider two instantiations of Theorem 4, based on the constructions of Ext-HPS given in [Wee10].

- The first construction is for the Diffie-Hellman relation $R_{\mathsf{DH}}((u, s), \alpha) = 1$ iff $u = s^\alpha$, with public parameters (g, g^α) where g is a generator for a group $\mathbb{G}$ of prime order q and $\alpha \leftarrow\!\!\$\, \mathbb{Z}_q$. Note that $\mathsf{SamR}(r) := (g^r, g^{\alpha r})$, for $r \leftarrow\!\!\$\, \mathbb{Z}_q$. The corresponding relation R'_{DH} is defined by $R'_{\mathsf{DH}}((u, s), r) = 1$ iff $u = g^r$ and $s = (g^\alpha)^r$; furthermore $f(r) = f_{g,\alpha}(r) := g^{\alpha r}$.

- The second construction is based on factoring. Let R_{QR} be defined by $R_{\mathsf{QR}}((u,s),k) = 1$ iff $u = s^{2^k}$, such that $(u,s) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+$, and with public parameters (N, g, g^{2^k}) where N is a $2k$-bit Blum integer and $g \leftarrow\!\!\$\ \mathbb{QR}_N^+$. Note that $\mathsf{SamR}(r) := (g^{2^k r}, g^r)$, for $r \leftarrow\!\!\$\ \mathbb{QR}_N^+$. The corresponding relation R'_{QR} is defined by $R'_{\mathsf{QR}}((u,s),r) = 1$ iff $u = (g^{2k})^r$ and $s = g^r$; furthermore $f(r) = f_g(r) := g^r$.

4.3 PAoK for Random Self-Reducible Languages

We construct a PAoK for languages that are random self-reducible. Random self-reducibility is a very natural property, with many applications in cryptography (see, e.g., [AL83, TW87, OO89]).

Random self-reducibility. Informally a function is random self-reducible if, given an algorithm that computes the function on random inputs, one can compute the function on any input. When considering *NP* relations, one has to take a little more care while defining random self-reducibility. We say that $\mathcal{O}_R(\cdot)$ is an *oracle* for the relation R, if on any input $x \in L_R$ we have that $(x, \mathcal{O}_R(x)) \in R$.

Definition 5 (Self-Reducible Relation)**.** An *NP*-relation R for a language L is random self-reducible if there exists a pair of PPT algorithms $\mathcal{W} := (\mathcal{W}_0, \mathcal{W}_1)$ such that, for any oracle $\mathcal{O}_R$ for the relation R, the following holds.

- For any $x \in L$, we have that $(x, w) \in R$ where w is defined as follows:
  - Let $x' := \mathcal{W}_0(x; r)$ for $r \leftarrow\!\!\$\ \{0,1\}^{poly(|x|)}$, and set $w' := \mathcal{O}_R(x')$;
  - Let $w := \mathcal{W}_1(x, w'; r)$;
- The value x' is uniformly distributed over L.

We call the pair of algorithms $\mathcal{W}$ an average-to-worst-case (AWC) reduction.

Notice that the reduction $\mathcal{W}$ has access to a "powerful" oracle that produces a witness for a randomized instance, and uses such witness to compute a witness for the original instance. As an example, consider the discrete logarithm problem in a cyclic group $\mathbb{G}$ of prime order q and with generator g. Given an instance x and an oracle $\mathcal{O}_{\mathsf{DLOG}}$ one can find w such that $g^w = x$ as follows: (i) Pick a random $r \in \mathbb{Z}_q$, compute $x' = x \cdot g^r$ and ask to the oracle $\mathcal{O}_{\mathsf{DLOG}}$ a witness for x'; (ii) Given w' such that $g^{w'} = x'$ compute $w := w' - r$.

Notice that, in the above example, given w and the auxiliary information r, one can easily compute a valid witness w' for the instance x'. This feature inspires the following property of a random self-reducible relation R:

Definition 6 (Witness Re-constructibility)**.** A random self-reducible relation R with AWC reduction $\mathcal{W}$ is *witness reconstructible* if there exists a PPT algorithm $\mathcal{W}_{\mathsf{inv}}$ such that for any $r \in \{0,1\}^{poly(|x|)}$ and for any $(x, w) \in R$ the following holds: Let x' be the oracle call made by $\mathcal{W}(x; r)$, and define $w' := \mathcal{W}_{\mathsf{inv}}(x, w; r)$; then $(x', w') \in R$.

The protocol. We show how to use a PAoK w.r.t. a specific sampler for a random self-reducible relation R, to construct a fully-extractable PAoK for the same relation. The idea is to map the input instance x into a random instance x', and to additionally send the prover the auxiliary information needed to compute a valid witness w' for x'. This way a honest prover essentially behaves as an oracle for the underlying relation R.

Let R be a random self-reducible *NP*-relation which is witness reconstructible and has AWC reduction $\mathcal{W} = (\mathcal{W}_0, \mathcal{W}_1)$. Let $\Pi' := (\mathsf{Chall}', \mathsf{Resp}')$ be a PAoK for R. Consider the following protocol $\Pi = (\mathsf{Chall}, \mathsf{Resp})$:

1. Upon input $1^\kappa, x$ algorithm Chall returns $c := (c', x', r)$ and b such that $x' = \mathcal{W}_0(x; r)$ (for $r \leftarrow_\$ \{0,1\}^{poly(|x|)}$) and $(c', b) \leftarrow_\$ \mathsf{Chall}(1^\kappa, x')$.
2. Upon input $1^\kappa, x, w, (c', x', r)$ algorithm Resp returns a where $a \leftarrow_\$ \mathsf{Resp}(1^\kappa, x', w', c')$ for $w' := \mathcal{W}_{\mathsf{inv}}(x, w; r)$.

Theorem 5. *Let R be a random self-reducible NP-relation which is witness reconstructible and has AWC reduction $\mathcal{W} = (\mathcal{W}_0, \mathcal{W}_1)$. Let Π' be a $(\mathcal{W}_0, \epsilon)$-PAoK for the relation R. Then protocol Π is an ϵ-PAoK for R.*

Before coming to the formal proof, let us discuss some intuition. Given a prover $\mathcal{P}^*$ for Π we need to define a knowledge extractor $\mathcal{K}$. The point is that $\mathcal{P}^*$ can equivalently be seen as a prover for Π' where instances are sampled using $\mathcal{W}_0(x; \cdot)$. For this scenario the knowledge soundness of Π provides a knowledge extractor $\mathcal{K}'$, and such an extractor can output a valid witness for a uniformly sampled instance. This is where we use the random self-reducibility property. The extractor $\mathcal{K}'$, in fact, can be seen as an oracle for the relation R that with noticeable probability produces a valid witness for a uniformly chosen instance. Therefore, using the AWC reduction $\mathcal{W}$ with oracle access to $\mathcal{K}'$ we can reconstruct a valid witness for the instance x.

Proof of Theorem 5. For any PPT prover $\mathcal{P}^*$ we need to define a knowledge extractor $\mathcal{K}$ such that, for any instance x and auxiliary input z_P for which $p(\kappa) := \Pr[\langle \mathcal{P}^*(1^\kappa, x, z), \mathcal{V}(1^\kappa, x)\rangle = \mathtt{acc}] > \epsilon(\kappa)$, the extractor $\mathcal{K}$ produces a witness w for x with probability $p(\kappa) - \epsilon(\kappa)$.

Let $\mathcal{K}'$ be the knowledge extractor w.r.t. $\mathcal{W}_0$ of Π'. Consider the knowledge extractor $\mathcal{K}$ that works as follow:

1. Pick a random $r \leftarrow_\$ \{0,1\}^{poly(\kappa)}$.
2. Compute $w' \leftarrow_\$ \mathcal{K}'(1^\kappa, x, z_P)$ and let $x' = \mathcal{W}_0(x; r)$.
3. If $(x', w') \in R$ then output $w := \mathcal{W}_1(x, w'; r)$, otherwise output $\perp$.

Clearly, the probability of $\mathcal{K}$ outputting $\perp$ is the same as $\mathcal{K}'$ outputting an invalid witness on a random instance. Hence:

$$\Pr_{\mathcal{K}}\left[(x, w) \in R : w \leftarrow_\$ \mathcal{K}(1^\kappa, x, z_P)\right] \geqslant p(\kappa) - \epsilon(\kappa).$$

This finishes the proof. □

4.4 PAoK for a Specific Sampler

We use the framework for obfuscation proposed by Bellare *et al.* in [BST14]. A circuit sampling algorithm is a PPT algorithm $\mathcal{S} = \{\mathcal{S}_\kappa\}_{\kappa \in \mathbb{N}}$ whose output is distributed over $\mathcal{C}_\kappa \times \mathcal{C}_\kappa \times \{0,1\}^{p(\kappa)}$, for a class of circuit $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ and a polynomial p. We assume that for every $C_0, C_1 \in \mathcal{C}_\kappa$ it holds that $|C_0| = |C_1|$. Given any class of samplers $\mathbf{S}$ for a class of circuits $\mathcal{C}$ consider the following definition:

Definition 7 ($\mathbf{S}$-Obfuscator)**.** A PPT algorithm Obf is an $\mathbf{S}$-obfuscator for the parametrized collection of circuits $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ if the following requirements are met.

- **Correctness:** $\forall \kappa, \forall C \in \mathcal{C}_\kappa, \forall x : \Pr[C'(x) = C(x) :\ C' \leftarrow_\$ \mathsf{Obf}(1^\kappa, C)] = 1$.
- **Security:** For every sampler $\mathcal{S} \in \mathbf{S}$, for every PPT (distinguishing) algorithm $\mathcal{D}$, and every auxiliary inputs $z_D, z_S \in \{0,1\}^*$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that for all $\kappa \in \mathbb{N}$:

$$\Big|\Pr\left[\mathcal{D}(C', aux, z_D, z_S) = 1 : \begin{array}{c}(C_0, C_1, aux) \leftarrow_\$ \mathcal{S}(1^\kappa, z_S), \\ C' \leftarrow_\$ \mathsf{Obf}(1^\kappa, C_0)\end{array}\right] - \Pr\left[\mathcal{D}(C', aux, z_D, z_S) = 1 : \begin{array}{c}(C_0, C_1, aux) \leftarrow_\$ \mathcal{S}(1^\kappa, z_S), \\ C' \leftarrow_\$ \mathsf{Obf}(1^\kappa, C_1)\end{array}\right]\Big| \leqslant \nu(\kappa),$$

where the probability is over the coins of $\mathcal{S}$ and Obf.

Abusing the notation, given a circuit sampler $\mathcal{S}$, we say that Obf is an $\mathcal{S}$-obfuscator if it is an $\{\mathcal{S}\}$-obfuscator. It is easy to see that the above definition allows to consider various flavours of obfuscation as a special case (including indistinguishability and differing-input obfuscation [BGI+12]). In particular, we say that a circuit sampler is differing-input if for any PPT adversary $\mathcal{A}$ and any auxiliary input $z_S \in \{0,1\}^*$ there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that the following holds:

$$\Pr\left[C_0(x) \neq C_1(x) \;:\; \begin{array}{c}(C_0, C_1, aux) \leftarrow\!\!\$\; \mathcal{S}(1^\kappa, z_S)\\ x \leftarrow \mathcal{A}(C_0, C_1, aux, z_S)\end{array}\right] \leqslant \nu(\kappa).$$

Let $\mathbf{S}^{\mathrm{diff}}$ be the class of all differing-input samplers; it is clear that an $\mathbf{S}^{\mathrm{diff}}$-obfuscator is equivalent to a differing-input obfuscator.

Consider the following construction of a PAoK $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ for a relation R.

- Upon input $(1^\kappa, x)$ algorithm $\mathsf{Chall}(1^\kappa, x)$ outputs $c := \mathsf{Obf}(C_{x,b})$ where $b \leftarrow\!\!\$\; \{0,1\}^\kappa$ and $C_{x,b}$ is the circuit that hard-wires x and b and, upon input a value w, it returns b if and only if $(x, w) \in R$ (and $\perp$ otherwise).
- Upon input $(1^\kappa, x, w, c)$, algorithm $\mathsf{Resp}(1^\kappa, x, w, c)$ executes $a := c(w)$ and outputs a.

Given an arbitrary instance sampler $\mathcal{S}$, let $\mathsf{CS}[\mathcal{S}]$ be the circuit samplers that sample randomness $r' := r\|b$, execute $(x, aux) := \mathcal{S}(1^\kappa, z_S; r)$, and output the tuple $(C_{x,b}, C_{x,\perp}, aux\|b)$. We prove the following result .

Theorem 6. *Let $\mathcal{S}$ be an arbitrary instance sampler and $\mathbf{S}^{\mathrm{diff}}$ and $\mathsf{CS}[\mathcal{S}]$ be as above. If $\mathsf{CS}[\mathcal{S}] \in \mathbf{S}^{\mathrm{diff}}$ and Obf is a $\mathsf{CS}[\mathcal{S}]$-obfuscator, then the protocol Π described above is an $\mathcal{S}$-PAoK for the relation R.*

Proof. Suppose that Π is not an $\mathcal{S}$-PAoK, we prove that $\mathsf{CS}[\mathcal{S}] \in \mathbf{S}^{\mathrm{diff}}$ but the Obf is not a $\mathsf{CS}[\mathcal{S}]$-Obfuscator. This means there exists a PPT adversary $\mathcal{P}^*$, and a polynomial $p(\cdot)$ such that the following holds. For for any PPT extractor $\mathcal{K}$ and infinitely many values of $\kappa \in \mathbb{N}$ there exist auxiliary informations $z_P, z_S \in \{0,1\}^*$, randomness $r_P \in \{0,1\}^*$, and a negligible function $\nu : \mathbb{N} \to [0,1]$ for which:

$$\begin{aligned}&\Pr\left[a = b : \begin{array}{c}(x, aux) \leftarrow\!\!\$\; \mathcal{S}(1^\kappa, z_S),\\ (c, b) \leftarrow\!\!\$\; \mathsf{Chall}(1^\kappa, x), a \leftarrow \mathcal{P}^*(1^\kappa, c, aux, z_P; r_P)\end{array}\right]\\ =&\Pr\left[a = b : \begin{array}{c}(C_{x,b}, C_{x,\perp}, aux\|b) \leftarrow\!\!\$\; \mathsf{CS}[\mathcal{S}](1^\kappa, z_S),\\ c \leftarrow\!\!\$\; \mathsf{Obf}(1^\kappa, C_{x,b}), a \leftarrow\!\!\$\; \mathcal{P}^*(1^\kappa, c, aux, z_P; r_P)\end{array}\right] \geqslant p(\kappa)\end{aligned} \tag{2}$$

but

$$\Pr\left[(x, w) \in R_L : \begin{array}{c}(x, aux) \leftarrow\!\!\$\; \mathcal{S}(1^\kappa, z_S),\\ w \leftarrow\!\!\$\; \mathcal{K}(1^\kappa, z_P, r_P, z_S)\end{array}\right] \leqslant \nu(\kappa). \tag{3}$$

Consider now a modified algorithm Chall' that, upon input $(1^\kappa, x)$, samples $b \leftarrow\!\!\$\; \{0,1\}^\kappa$ as Chall would do, but then outputs $c := \mathsf{Obf}(C_{x,\perp})$. Obviously:

$$\begin{aligned}&\Pr\left[a = b : \begin{array}{c}(x, aux) \leftarrow\!\!\$\; \mathcal{S}(1^\kappa, z_S),\\ (c, b) \leftarrow\!\!\$\; \mathsf{Chall}'(1^\kappa, x), a \leftarrow\!\!\$\; \mathcal{P}^*(1^\kappa, c, aux, z_P; r_P)\end{array}\right]\\ =&\Pr\left[a = b : \begin{array}{c}(C_{x,b}, C_{x,\perp}, aux\|b) \leftarrow\!\!\$\; \mathsf{CS}[\mathcal{S}](1^\kappa, z_S),\\ c \leftarrow\!\!\$\; \mathsf{Obf}(1^\kappa, C_{x,\perp}), a \leftarrow\!\!\$\; \mathcal{P}^*(1^\kappa, c, aux, z_P; r_P)\end{array}\right] = 2^{-\kappa}\end{aligned} \tag{4}$$

It is not hard to see that Eq. (3) implies that $\mathsf{CS}[\mathcal{S}] \in \mathbf{S}^{\mathrm{diff}}$ and Eq. (2) and Eq. (4) imply that from $\mathcal{P}^*$ we can build a distinghuisher for Obf by checking that the equation $a = b$ holds. This concludes the proof. □

By combining Theorem 6 together with Theorem 5 we get the following corollary.

Corollary 1. *Let R be a random self-reducible NP-relation which is witness reconstructible and has AWC reduction $\mathcal{W} = (\mathcal{W}_{\mathsf{smp}}, \mathcal{W}_{\mathsf{cmp}}, \mathcal{W}_{\mathsf{inv}})$. If there exists a $\mathsf{CS}[\mathcal{W}_{\mathsf{smp}}]$-obfuscator and $\mathsf{CS}[\mathcal{W}_{\mathsf{smp}}] \in \mathbf{S}^{\mathrm{diff}}$ then there exists a PAoK for R.*

5 On Zero Knowledge

One can easily verify that PAoK are always honest-verifier zero-knowledge, since the answer to a (honest) challenge from the verifier can be predicted without knowing a valid witness.

It is also not too hard to see that in general PAoK may not be witness indistinguishable (more details in the full version of the paper [FNV15b]).

Furthermore, we note that PAoK in the plain model can be zero-knowledge only for trivial languages. The reason is that predictable arguments have inherently deterministic provers and, as shown by Goldreich and Oren [GO94, Theorem 4.5], the zero-knowledge property for such protocols is achievable only for languages in *BPP*.

In this section we show how to circumvent this impossibility using setup assumptions. In particular, we show how to transform any PAoK into another PAoK additionally satisfying the zero-knowledge property (without giving up on predictability). We provide two solutions. The first one in the common random string (CRS) model,[14] while the second one is in the non-programmable random oracle (NPRO) model.

5.1 Compiler in the CRS Model

We start by recalling the standard notion of zero-knowledge interactive protocols in the CRS model. Interactive protocols in the CRS model are defined analogously to interactive protocols in the plain model (cf. Section 2.3.1), with the only difference that at setup a uniformly random string $\omega \leftarrow\!\!\$\, \{0,1\}^\ell$ is sampled and both the prover and the verifier additionally take ω as input.

Definition 8 (Zero-knowledge protocol in the CRS model)**.** Let $(\mathcal{P}, \mathcal{V})$ be an interactive protocol for an *NP* relation R. We say that $(\mathcal{P}, \mathcal{V})$ satisfies the *zero-knowledge* property in the CRS model if for every PPT malicious verifier $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{Z} = (\mathcal{Z}_0, \mathcal{Z}_1)$ and a negligible function $\nu : \mathbb{N} \to [0,1]$ such that for all PPT distinguishers $\mathcal{D}$, all $(x,w) \in R$, and all auxiliary inputs $z \in \{0,1\}^*$, the following holds:

$$\Delta(\Pi, \mathcal{Z}, \mathcal{V}^*) := \max_{\mathcal{D},z} \Bigg| \Pr\left[\mathcal{D}(\omega, x, \tau, z) = 1 : \begin{array}{c} \omega \leftarrow\!\!\$\, \{0,1\}^\ell, \\ \tau \leftarrow (\mathcal{P}(\omega, x, w) \leftrightarrows \mathcal{V}^*(\omega, x, z)) \end{array}\right] - \Pr\left[\mathcal{D}(\omega, x, \tau, z) = 1 : \begin{array}{c} (\omega, \vartheta) \leftarrow\!\!\$\, \mathcal{Z}_0(1^\kappa), \\ \tau \leftarrow \mathcal{Z}_1(\vartheta, x, z) \end{array}\right] \Bigg| \leqslant \nu(|x|).$$

The compiler. Our first compiler is based on a NIZK-PoK system (see full version for the formal definition). Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be a PAoK for a relation R, and assume that Chall uses at most $\rho(|x|, \kappa)$ random bits for a polynomial ρ. Let $\mathcal{NIZK} = (\ell, \mathsf{Prove}, \mathsf{Ver})$ be a NIZK for the relation

$$R_{\mathsf{chal}} = \{((c,x), r) : \exists b \text{ s.t. } (c,b) := \mathsf{Chall}(1^\kappa, x; r)\}.$$

Consider the following one-round PAoK $\Pi' = (\mathsf{Chall}', \mathsf{Resp}')$ in the CRS model.

[14] This model is sometimes also known as the Uniform Random String (URS) model.

- At setup a uniform CRS $\omega \leftarrow\!\$\ \{0,1\}^\ell$ is sampled.
- Algorithm Chall' takes as input $(1^\kappa, \omega, x)$ and proceeds as follows:
  1. Sample random tape $r \leftarrow\!\$\ \{0,1\}^\rho$.
  2. Generate a proof $\pi \leftarrow\!\$\ \mathsf{Prove}(\omega, (c,x), r)$ for $((c,x), r) \in R_{\mathsf{chal}}$.
  3. Output $c' := (c, \pi)$.
- Algorithm Resp' takes as input $(1^\kappa, \omega, x, w, c')$ and proceeds as follows:
  1. Parse $c' := (c, \pi)$; in case $\mathsf{Ver}(\omega, (c,x), \pi) = 0$ return $\perp$.
  2. Output $b' := \mathsf{Resp}(1^\kappa, x, w, c)$.

Roughly speaking, in the above construction the verifier sends the challenge c together with a NIZK-PoK π that c is "well formed" (i.e., there exist random coins r such that the verifier of the underlying PAoK with coins r returns a pair (c, b)); the prover answers only in case the proof π is correct. We show the following result.

Theorem 7. *Let Π be a PAoK for the relation $R \in NP$ and let $\mathcal{NIZK}$ be a NIZK-PoK for the relation R_{chal}. Then the protocol Π' is a ZK-PAoK in the CRS model.*

The knowledge soundness of Π' follows almost directly from the zero-knowledge property of $\mathcal{NIZK}$ and from the knowledge soundness of Π. In fact, one can consider a mental experiment where the verifier generates a simulated proof π instead of a real one. This proof does not carry any information about the randomness but it is indistinguishable from a real one. A successful prover in the real world is still successful in this mental experiment and, therefore, we reduced to the knowledge soundness of Π. The zero-knowledge of Π' follows from the fact that PAoK are honest-verifier zero-knowledge, and from the knowledge soundness of $\mathcal{NIZK}$. In particular, given a maliciously generated challenge (c^*, π^*), the simulator can use the knowledge extractor of $\mathcal{NIZK}$ on π^*, extract a valid witness r^*, and then produce a valid answer.

Proof of Theorem 7. The proof of the completeness property follows readily from the completeness of the NIZK and of the underlying PAoK.

We proceed to show knowledge soundness of Π', by relying on the zero-knowledge property of the NIZK. Let $\mathcal{P}^*$ be the prover in the definition of knowledge soundness, and consider the hybrid experiment where at setup we run $(\omega, \vartheta) \leftarrow\!\$\ \mathcal{Z}_0(1^\kappa)$ and we replace a protocol transcript with $\tau \leftarrow\!\$\ \mathcal{Z}_1(\vartheta, x)$. By unbounded[15] zero-knowledge of the NIZK, it follows that the succees probability of $\mathcal{P}^*$ is negligibly close in the two experiments. We can thus define the knowledge extractor of Π' to be the same as the one for Π, additionally executing the simulated hybrid experiment described above. Knowledge soundness of Π' follows.

Next, we turn to show the zero-knowledge property. Consider the simulator $\mathcal{Z}' = (\mathcal{Z}'_0, \mathcal{Z}'_1)$ described below.

- Algorithm $\mathcal{Z}'_0$ returns $(\omega, \vartheta) \leftarrow\!\$\ \mathcal{K}_0(1^\kappa)$.
- Algorithm $\mathcal{Z}'_1$ first runs the malicious verifier in order to obtain a challenge $c' \leftarrow\!\$\ \mathcal{V}^*(\omega, x, z)$ where $c' := (c, \pi)$. It then checks whether $\mathsf{Ver}(\omega, (c,x), \pi) = 0$, in which case it outputs a simulated transcript $\tau := (c', \perp)$. Otherwise, in case the proof is accepting, it attempts to extract a witness $r \leftarrow\!\$\ \mathcal{K}_1(\omega, \vartheta, (c,x), \pi)$; in case the extractor fails, $\mathcal{Z}'_1$ outputs a simulated transcript $\tau := (c', \perp)$, and otherwise it runs $(c, b) \leftarrow\!\$\ \mathsf{Chall}(1^\kappa, x; r)$ and returns $\tau := (c', b)$.

To show that the above simulation strategy is sound, we rely on both the honest-verifier zero-knowledge property of Π and on the knowledge soundness of the NIZK.

[15] Strictly speaking, for this step of the proof to go through, single-theorem zero-knowledge (see [SP92]) would actually be sufficient.

Let $T_r = (\omega, c' := (c, \pi), b)$ be the random variable corresponding to a transcript produced by an interaction between $\mathcal{V}^*$ and the honest prover $\mathcal{P}$. Similarly, denote with $T_s = (\tilde{\omega}, \tilde{c}' := (\tilde{c}, \tilde{\pi}), \tilde{b})$ the random variable corresponding to a transcript produced by the above described simulator $\mathcal{Z}'$. We will show that T_r and T_s are close in statistical distance, which concludes the proof. Consider the following events:

$$A := \{\mathsf{Ver}(\tilde{\omega}, \tilde{c}, \tilde{\pi}) = 0 \ \vee \ ((\tilde{c}, x), r) \in R_{\mathsf{chal}}\}$$
$$B := \left\{(\tilde{\omega} = \omega \wedge \tilde{c}' = c') \Rightarrow \tilde{b} = b\right\}.$$

It is easy to verify that there exists a negligible function $\nu : \mathbb{N} \to [0, 1]$ such that $\Delta(T_r, T_s|A \wedge B) \leq \nu(\kappa)$. This is because either the verification of π fails (and therefore the response is $\perp$ both for real and simulated transcripts), or the extraction succeeds and so the simulator $\mathcal{Z}_1$ produces an answer $\tilde{b} \neq \perp$ and, for a fixed challenge and CRS, the event B ensures that we get the same answer as in the real distribution. Notice that the challenges in both the real and simulated transcripts are derived in the same way as a (randomized) function of the CRS; therefore the distributions of the pairs (ω, c') and $(\tilde{\omega}, \tilde{c}')$ are negligible close (since ω and $\tilde{\omega}$ are negligible close).

By Lemma 1 , (cf. Section 2), and applying a union bound, we have that:

$$\Delta(T_r, T_s) \leqslant \Pr[\neg A] + \Pr[\neg B] + \nu(\kappa).$$

So we are left with bounding the probability of events A and B not happening. To bound the probability of $\neg A$ we rely on knowledge soundness. In particular, there exists a negligible function $\nu' : \mathbb{N} \to [0, 1]$ such that

$$\begin{aligned} \Pr[\neg A] &= \Pr[\mathsf{Ver}(\tilde{\omega}, \tilde{c}, \tilde{\pi}) = 1 \wedge ((\tilde{c}, x), r) \notin R_{\mathsf{chal}}] \\ &\leqslant \Pr[\mathsf{Ver}(\tilde{\omega}, \tilde{c}, \tilde{\pi}) = 1] - \Pr[((\tilde{c}, x), r) \in R_{\mathsf{chal}}] \leqslant \nu'(\kappa). \end{aligned}$$

The fact that the probability of $\neg B$ is negligible follows readily by completeness of Π. □

5.2 Compiler in the NPRO Model

We start by recalling the definition of zero-knowledge in the NPRO model, for interactive protocols. Recall that a NPRO is weaker than a programmable random oracle. Intuitively, in the NPRO model the simulator can observe the verifier's queries to the hash function, but is not allowed to program the behaviour of the hash function. The definition below is adapted from Wee [Wee09].

Definition 9 (Zero-knowledge protocol in the NPRO model)**.** Let $(\mathcal{P}, \mathcal{V})$ be an interactive protocol for an *NP* relation R. We say that $(\mathcal{P}, \mathcal{V})$ satisfies the *zero-knowledge* property in the NPRO model if for every PPT malicious verifier $\mathcal{V}^*$ there exists a PPT simulator $\mathcal{Z}$ and a negligible function $\nu : \mathbb{N} \to [0, 1]$ such that for all PPT distinguishers $\mathcal{D}$, all $(x, w) \in R$, and all auxiliary inputs $z \in \{0, 1\}^*$, the following holds:

$$\begin{aligned} \Delta(\Pi, \mathcal{Z}, \mathcal{V}^*) := \max_{\mathcal{D}, z} \Big| &\Pr\left[\mathcal{D}^H(x, \tau, z) = 1 : \ \tau \leftarrow (\mathcal{P}^H(x, w) \leftrightarrows \mathcal{V}^{*H}(x, z))\right] \\ &- \Pr\left[\mathcal{D}^H(x, \tau, z) = 1 : \ \tau \leftarrow \mathcal{Z}^H(x, z)\right]\Big| \leqslant \nu(|x|). \end{aligned}$$

The compiler. Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be a PAoK for a relation R with ℓ-bit prover's answer, and assume that Chall uses at most $\rho(|x|, \kappa)$ random bits for a polynomial ρ. Let H be a random oracle with output length $\rho(\kappa)$. Consider the following derived one-round PAoK $\Pi' = (\mathsf{Chall}', \mathsf{Resp}')$.

- Algorithm Chall' takes as input $(1^\kappa, x)$ and proceeds as follows:
  1. Sample a random tag $t_1 \leftarrow\$ \{0,1\}^\rho$ and compute $r := H(t_1)$.
  2. Run $(c, b) := \mathsf{Chall}(1^\kappa, x; r)$.
  3. Define $t_2 := H(b)$, and set the challenge to $c' := (c, t)$ where $t := t_1 \oplus t_2$.
- Algorithm Resp' takes as input (x, w, c) and proceeds as follows:
  1. Parse $c' := (c, t)$ and run $a \leftarrow\$ \mathsf{Resp}(1^\kappa, x, w, c)$.
  2. Define $t_1 := t \oplus H(a)$, and check whether $(c, a) = \mathsf{Chall}(1^\kappa, x; H(t_1))$. If this is the case, output a and otherwise output $\perp$.

The main idea behind the above construction is to force the malicious verifier to follow the underlying protocol Π; in order to do so we generate the challenge feeding the algorithm Chall with the uniformly random string $H(t_1)$. What we need now is to both make able the prover to check that the verifier followed the algorithm Chall and to maintain soundness. Unfortunately, since PAoK are private-coin protocols, we can't simply make the verifier output t_1; what we do instead is to one-time pad the value with the value t_2 which is computable only knowing the answer. We show the following result:

Theorem 8. *If Π is a PAoK with ℓ-bit prover's answer for the relation R, and $\ell = \omega(\log \kappa)$, then the protocol Π' is a ZK-PAoK in the NPRO model.*

Before coming to the proof, let us discuss some intuition. To prove soundness we show that $t = t_1 \oplus t_2$ is essentially uniformly random if the prover does not know b: this explains why we need $\ell = \omega(\log \kappa)$, otherwise a malicious prover could just brute force the right value of b and check for consistency. Note that here we are leveraging the power of the random oracle model, that allows us to produce polynomially-long pseudorandomness from unpredictability. To prove zero-knowledge we note that a simulator can look into the random-oracle calls made by the malicious verifier while running it. Given the output (c^*, t^*) produced by the malicious verifier two cases can happen:

- The simulator finds an oracle call t' that "explains" the challenge c^*, namely $(c^*, b) = \mathsf{Chall}(1^\kappa, x; H(t'))$; in this case the simulator just outputs b. We argue that the simulator produces an indistinguishable view because the protocol Π has overwhelming completeness.
- The simulator does not find any t' that explains the challenge. Then it outputs $\perp$. Let b' be the answer that the real prover would compute using the algorithm Resp. We argue that the malicious verifier can find a challenge (c^*, t^*) that passes the check, namely $(c^*, b') = \mathsf{Chall}(1^\kappa, x; H(H(b') \oplus t^*))$ only with negligible probability. Therefore the real prover would output $\perp$ as well, and so the views are indistinguishable.

Proof of Theorem 8. Completeness follows readily from the completeness of the underlying PAoK.

We proceed to prove knowledge soundness of Π'. Given a prover $\mathcal{P}'^*$ for Π' that makes the verifier accept with probability $p(\kappa)$, we define a prover $\mathcal{P}^*$ for Π that is successful with probability $p(\kappa)/Q(\kappa)$ where Q is a polynomial that upper bounds the number of oracle calls made by $\mathcal{P}'^*$ to the NPRO H. Prover $\mathcal{P}^*$ proceeds as follow:

1. Upon input $(1^\kappa, c, z)$, set $c' := (c, t)$ for uniformly random $t \leftarrow\$ \{0,1\}^\rho$ and run $\mathcal{P}^*(1^\kappa, c', z)$. Initialize counter j to $j := 1$, $\mathcal{Q} := \emptyset$, and pick a uniformly random index $i^* \leftarrow\$ [Q(\kappa)]$.

2. Upon input a random oracle query x from $\mathcal{P}'^*$, pick $y \leftarrow\!\!\$\ \{0,1\}^\rho$ and add the tuple (x, y, j) to H. If $j = i^*$, then output x and stop. Otherwise set $j \leftarrow j + 1$ and forward y to $\mathcal{P}'^*$.
3. In case $\mathcal{P}^*$ aborts or terminates, output $\bot$ and stop.

Without loss of generality we can assume that the prover $\mathcal{P}'^*$ does not repeat random oracle queries, and that before outputting an answer a^*, it checks that $(c, a^*) := \mathsf{Chall}(1^\kappa, x; H(t \oplus H(a^*)))$. We now analyse the winning probability of $\mathcal{P}^*$. Let a be the correct answer corresponding to the challenge c. Observe that the view produced by $\mathcal{P}^*$ is exactly the same as the real view (i.e., the view that $\mathcal{P}'^*$, with access to the random oracle, expects from an execution with the verifier $\mathcal{V}'$ from Π'), until $\mathcal{P}'^*$ queries H with the value a. In this case, in fact, $\mathcal{P}'^*$ expects to receive a tag t_2 such that $(c, a) := \mathsf{Chall}(1^\kappa, x; H(t \oplus t_2))$. We can write,

$$\begin{aligned}
\Pr&\big[\mathcal{P}^*(1^\kappa, c, z) \text{ returns } a\big] \\
&= \Pr\big[(a, *, i^*) \in \mathcal{Q}\big] \\
&= \Pr\big[a \text{ is the } i^*\text{-th query to } H \ \wedge\ a = \mathcal{P}'^*(1^\kappa, c', z)\big] \\
&= \Pr\big[a \text{ is the } i^*\text{-th query to } H \mid a = \mathcal{P}^*(1^\kappa, c', z)\big] \Pr\big[a = \mathcal{P}^*(1^\kappa, c', z)\big] \\
&\geqslant 1/Q(\kappa) \cdot p(\kappa).
\end{aligned} \tag{5}$$

Notice that in Eq.(5) the two probabilities are taken over two different probability spaces, namely the view provided by $\mathcal{P}'^*$ to the prover $\mathcal{P}^*$ together with i^* on the left hand side and the view that $\mathcal{P}'^*$ would expect in an execution with a honest prover together with the index i^* in the right hand side. Knowledge soundness of Π' follows.

We now prove the zero-knowledge property. Upon input $(1^\kappa, x, z)$ the simulator $\mathcal{Z}$ proceeds as follows:

1. Execute algorithm $\mathcal{V}^*(1^\kappa, x, z)$ and forward all queries to H; let $\mathcal{Q}$ be the set of queries made by $\mathcal{V}^*$.
2. Eventually $\mathcal{V}^*$ outputs a challenge $c^* = (c'^*, t^*)$. Check if there exist $(a^*, t_1^*) \in \mathcal{Q}$ such that $(c'^*, a^*) = \mathsf{Chall}(1^\kappa, x; H(t_1^*))$ and $t^* = t_1^* \oplus H(a^*)$. Output the transcript $\tau := (c^*, a^*)$. If no such pair is found, output $(c^*, \bot)$.

Let r' be the randomness used by the prover. For any challenge c, instance x and witness w, we say that r is *good* for c w.r.t. x, w, r' if $(c, a) = \mathsf{Chall}(1^\kappa, x; r) \wedge a = \mathsf{Resp}(1^\kappa, x, w, c; r')$. By completeness, the probability that r is not good, for $r \leftarrow\!\!\$\ \{0,1\}^\rho$, is negligible. Therefore by letting *Good* be the event that $\mathcal{V}^*$ queries H only on inputs that output good randomness for some c, by taking a union bound over all queries we obtain

$$\Pr[\mathit{Good}] \geqslant 1 - Q(\kappa) \cdot \nu'(\kappa) \geqslant 1 - \nu(\kappa), \tag{6}$$

for negligible functions $\nu, \nu' : \mathbb{N} \to [0, 1]$.

From now on we assume that the event *Good* holds; notice that this only modifies by a negligible factor the distinguishing probability of the distinguisher $\mathcal{D}$.

We proceed with a case analysis on the possible outputs of the simulator and the prover:

- The second output of $\mathcal{Z}$ is $a \neq \bot$, whereas the second output of $\mathcal{P}$ is $\bot$. Conditioning on $\mathcal{Z}$'s second output being $a \neq \bot$, we get that the challenge c is *well formed*, namely, c is in the set of all possible challenges for the instance x and security parameter 1^κ. On the other hand, the fact that $\mathcal{P}$ outputs $\bot$ means that either algorithm Resp aborted or the check in step 2 of the description of Π' failed. However, neither of the two cases can happen unless event *Good* does not happen. Namely, if Resp outputs $\bot$ the randomness $H(t^* \oplus H(a))$ is not good for c (w.r.t. x, w, r'), and therefore Resp must have output a which, together with t^*, would pass the test in step 2 by definition of $\mathcal{Z}$. It follows that this case happens only with negligible probability.

- The second output returned by $\mathcal{Z}$ is $\bot$, whereas $\mathcal{P}$'s second output is $a \neq \bot$. Conditioning on $\mathcal{Z}$'s second output being $\bot$, we get that $\mathcal{V}^*$ made no queries (a^*, t_1^*) such that $(c, a^*) = \mathsf{Chall}(1^\kappa, x; t_1^*)$ and $t_1^* = H(t^* \oplus H(a^*))$. In such a case, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:

$$\begin{aligned}&\Pr[(c,a) = \mathsf{Chall}(1^\kappa, x; H(t^* \oplus H(a^*)))]\\ &\leqslant \Pr\left[t_1^* := (H(a) \oplus t) \in \mathcal{Q} \vee \mathsf{Chall}(1^\kappa, x; H(t_1^*)) = (c,a)\right]\\ &\leqslant Q \cdot 2^{-\rho} + 2^{-\gamma} + \epsilon \leqslant \nu(\kappa),\end{aligned} \tag{7}$$

where 2^γ is the size of the challenge space. Notice that by overwhelming completeness and $\ell = \omega(\log \kappa)$, it follows that $\gamma = \omega(\log \kappa)$.

- Both $\mathcal{Z}$'s and $\mathcal{P}$'s second output are not $\bot$, but they are different. This event cannot happen, since we are conditioning on *Good*.

Combining Eq.(6) and Eq.(7) we obtain that $\Delta(\Pi, \mathcal{Z}, \mathcal{V}^*)$ is negligible, as desired. □

On RO-dependent auxiliary input. Notice that Definition 9 does not allow the auxiliary input to depend on the random oracle. Wee [Wee09] showed that this is necessary for one-round protocols, namely zero-knowledge w.r.t. RO-dependent auxiliary input is possible only for trivial languages. This is because the result of [GO94] relativizes.

In a similar fashion, for the case of multi-round protocols, one can show that also the proof of [GO94, Theorem 4.5] relativizes. It follows that the assumption of disallowing RO-dependent auxiliary input is necessary also in our case.

6 Predictable ZAPs

We recall the concept of ZAP introduced by Dwork and Naor [DN07]. ZAPs are two-message (i.e., one-round) protocols in which:

(i) The first message, going from the verifier to the prover, can be fixed "once and for all," and is independent of the instance being proven;
(ii) The verifier's message consists of public coins.

Typically a ZAP satisfies two properties. First, it is witness indistinguishable meaning that it is computationally hard to tell apart transcripts of the protocols generated using different witnesses (for a given statement). Second, the protocol remains sound even if the statement to be proven is chosen after the first message is fixed.

In this section we consider the notion of Predictable ZAP (PZAP). With the terminology "ZAP" we want to stress the particular structure of the argument system we are interested in, namely a one-round protocol in which the first message can be fixed "once and for all." However, there are a few important differences between the notion of ZAPs and PZAPs. First off, PZAPs cannot be public coin, because the predictability requirement requires that the verifier uses private coins. Second, we relax the privacy requirement and allow PZAPs not to be witness indistinguishable; notice that, in contrast to PZAPs, ZAPs become uninteresting in this case as the prover could simply forward the witness to the verifier. Third, ZAPs are typically only computationally sound, whereas we insist on knowledge soundness.

More formally, a PZAP is fully specified by a tuple of PPT algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{Predict})$ as described below:

1. $\mathcal{V}$ samples $(c, \vartheta) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa)$ and sends c to $\mathcal{P}$.

2. $\mathcal{P}$ samples $a \leftarrow\$ \mathsf{Resp}(1^\kappa, x, w, c)$ and sends a to $\mathcal{V}$.
3. $\mathcal{V}$ computes $b := \mathsf{Predict}(1^\kappa, \vartheta, x)$ and outputs $\mathtt{acc}$ iff $a = b$.

Notice that, in contrast to the syntax of PAoK, now the verifier runs two algorithms $\mathsf{Chall}, \mathsf{Predict}$, where Chall is independent of the instance x being proven, and $\mathsf{Predict}$ uses the trapdoor ϑ and the instance x in order to predict the prover's answer.

Care needs to be taken while defining (knowledge) soundness for PZAPs. In fact, observe that while the verification algorithm needs private coins, in many practical circumstances the adversary might be able to infer the outcome of the verifier, and thus learn one bit of information about the verifier's private coins. For this reason, as we aim to constructing argument systems where the first message can be re-used, we enhance the adversary with oracle access to the verifier in the definition of soundness.

Definition 10 (Predictable ZAP)**.** Let $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{Predict})$ be as specified above, and let R be an *NP* relation. Consider the properties below.

Completeness: There exists a negligible function $\nu : \mathbb{N} \to [0, 1]$ such that for all $(x, w) \in R$:

$$\Pr_{c,\vartheta}\left[\mathsf{Predict}(1^\kappa, \vartheta, x) \neq \mathsf{Resp}(1^\kappa, x, w, c) : \ (c, \vartheta) \leftarrow \mathsf{Chall}(1^\kappa)\right] \leq \nu(\kappa).$$

(Adaptive) Knowledge soundness with error ϵ: For all PPT provers $\mathcal{P}^*$ making polynomially many queries to its oracle, there exists a PPT extractor $\mathcal{K}$ such that for any auxiliary input $z \in \{0, 1\}^*$ the following holds. Whenever

$$p_z(\kappa) := \Pr\left[a = b \ : \ \begin{array}{l}(c, \vartheta) \leftarrow\$ \mathsf{Chall}(1^\kappa),\\ (x, a) \leftarrow\$ \mathcal{P}^{*\mathcal{V}(1^\kappa,\vartheta,\cdot,\cdot)}(c, z) \text{ where } |x| = \kappa,\\ b := \mathsf{Predict}(1^\kappa, \vartheta, x).\end{array}\right] > \epsilon(\kappa),$$

we have

$$\Pr\left[(x, w) \in R : \ \begin{array}{l}(c, \vartheta) \leftarrow\$ \mathsf{Chall}(1^\kappa),\\ (x, a) \leftarrow\$ \mathcal{P}^{*\mathcal{V}(1^\kappa,\vartheta,\cdot,\cdot)}(c, z) \text{ where } |x| = \kappa,\\ w \leftarrow\$ \mathcal{K}(1^\kappa, x, z, \mathcal{Q}).\end{array}\right] \geqslant p_z(\kappa) - \epsilon(\kappa).$$

In the above equations, we denote by $\mathcal{V}(1^\kappa, \vartheta, \cdot, \cdot)$ the oracle machine that upon input a query (x, a) computes $b := \mathsf{Predict}(1^\kappa, \vartheta, x)$ and outputs 1 iff $a = b$; we also write $\mathcal{Q}$ for the list $\{((x_i, a_i), d_i)\}$ of oracle queries (and answers to these queries) made by $\mathcal{P}^*$.

Let ℓ be the size of the prover's answer, we call Π a predictable ZAP (PZAP) for R if Π satisfies completeness and adaptive knowledge soundness with error ϵ, and moreover $\epsilon - 2^{-\ell}$ is negligible. In case knowledge soundness holds provided that no verification queries are allowed, we call Π a *weak* PZAP.

The definition of *laconic* PZAPs is obtained as a special case of the above definition by setting $\ell = 1$. Note, however, that in this case we additionally need to require that the value x returned by $\mathcal{P}^*$ is not contained in $\mathcal{Q}$.[16]

[16] This is necessary, as otherwise a malicious prover could query both $(x, 0)$ and $(x, 1)$, for $x \notin L$, and succeed with probability 1.

6.1 Construction via Extractable Witness PRF

We provide a construction of a PZAP based on any extractable witness pseudo-random function (Ext-WPRF), a primitive recently introduced in [Zha16]. An Ext-WPRF is a tuple of algorithms $\Pi_{\mathsf{wprf}} := (\mathsf{KGen}, \mathsf{F}, \mathsf{Eval})$ specified as follow.

- Upon input the security parameter, and a relation R, the key generation algorithm KGen returns a public evaluation key ek and a secret key fk.
- Upon input fk and an instance $x \in \{0,1\}^\kappa$, algorithm F produces an output y.
- Upon input (ek, x, w) the public evaluation algorithm Eval returns the value $\mathsf{F}(fk, x)$ if w is a valid witness for x, and $\perp$ otherwise.

We say that Π_{wprf} is complete if $\mathsf{F}(fk, x) = \mathsf{Eval}(ek, x, w)$ for all $(x, w) \in R$ and all (ek, fk) returned by KGen. Consider the following experiment $\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(b, \kappa)$ parametrized by an *NP* relation R, an adversary $\mathcal{A}$, the security parameter κ, and a bit b:

1. Run $(ek, fk) \leftarrow\$ \mathsf{KGen}(1^\kappa, R)$.
2. Upon input ek and auxiliary input $z \in \{0,1\}^*$, the adversary $\mathcal{A}$ can adaptively make polynomially many queries (so-called F-queries) on instances $x_i \in \{0,1\}^\kappa$, to which the challenger answers by returning $\mathsf{F}(fk, x_i)$.
3. Eventually, $\mathcal{A}$ can make a single challenge query by specifying an instance $x^* \in \{0,1\}^\kappa$. Upon input such a query, the challenger computes $y_0 := \mathsf{F}(fk, x^*)$ and $y_1 \leftarrow\$ \{0,1\}^\kappa$ and returns y_b to $\mathcal{A}$.
4. After making additional F-queries, $\mathcal{A}$ produces a bit b'. Hence, the challenger checks that x^* was not part of any F-query made by $\mathcal{A}$. If this is not the case, the experiment returns a random bit. Otherwise, it returns b'.

We call $\alpha_{\mathsf{wprf}}(R, \mathcal{A}) := \left| \Pr[\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(0, \kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(1, \kappa) = 1] \right|$ the advantage of $\mathcal{A}$ in the above experiment.

Consider, additionally, the experiment $\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A},\mathcal{K}}(\kappa)$ parametrized by an *NP* relation R, an adversary $\mathcal{A}$, an extractor $\mathcal{K}$, and the security parameter κ.

1. Run $(ek, fk) \leftarrow\$ \mathsf{KGen}(1^\kappa, R)$;
2. Upon input ek and auxiliary input $z \in \{0,1\}^*$, adversary $\mathcal{A}$ can adaptively make polynomially many queries (so-called F-queries) on instances $x_i \in \{0,1\}^\kappa$, to which the challenger answers by returning $\mathsf{F}(fk, x_i)$.
3. Eventually, $\mathcal{A}$ can make a single challenge query by specifying an instance $x^* \in \{0,1\}^\kappa$. Upon input such a query, the challenger returns $y^* := \mathsf{F}(fk, x^*)$.
4. Let $\mathcal{Q} = \{(x_i, y_i)\}$ be the set of all F-queries (and corresponding answers) made by $\mathcal{A}$. The challenger runs $w^* \leftarrow\$ \mathcal{K}(ek, x^*, y^*, z, \mathcal{Q})$ and the experiment returns 1 iff $(x^*, w^*) \in R$.

Intuitively, security of an Ext-WE requires that for all adversaries that can distinguish the output of F from random (as defined in the first experiment) with noticeable probability, there exists an extractor that can extract a valid witness for the value x^* returned by the adversary with noticeable probability. For our purpose, we use a slightly stronger formulation where the definition holds for non-uniform polynomial-time adversaries and, as done before, the extraction probability match the advantage of the adversary. We note, however, that the constructions from [Zha16] are secure under the non-uniform formulation by considering similar strengthening on the underlying assumptions.

Definition 11 (Extractable witness PRF)**.** We say that $\Pi_{\mathsf{wprf}} = (\mathsf{KGen}, \mathsf{F}, \mathsf{Eval})$ is secure[17] for a relation R if, for all PPT adversaries $\mathcal{A}$ such that $\alpha_{\mathsf{wprf}}(R, \mathcal{A}) \geqslant 1/p(\kappa)$ for a non-zero polynomial $p(\cdot)$, there exists a PPT extractor $\mathcal{K}$ such that $\Pr[\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A},\mathcal{K}}(\kappa) = 1] \geqslant 1/p(\kappa)$.

[17] This is called adaptive instance interactive security in [Zha16].

The construction. Let $\Pi_{\mathsf{wprf}} = (\mathsf{KGen}, \mathsf{F}, \mathsf{Eval})$ be an Ext-WPRF for an *NP* relation R. Consider the following construction of a PZAP $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{Predict})$ for the same relation.

- Upon input 1^κ, define $\mathsf{Chall}(1^\kappa) := \mathsf{KGen}(1^\kappa, R)$.
- Upon input $(1^\kappa, x, w, c := ek)$, define $\mathsf{Resp}(1^\kappa, x, w, c) := \mathsf{Eval}(ek, x, w)$.
- Upon input $(\vartheta := fk, x)$, define $\mathsf{Predict}(\vartheta, x) := \mathsf{F}(fk, x)$.

Theorem 9. *Assume that* Π_{WPRF} *is a secure Ext-WPRF for an NP relation* R*. Then* Π *as defined above is a PZAP for the same relation* R*.*

Proof. By contradiction, assume that Π is not a PZAP, namely there exists a prover $\mathcal{P}^*$, an auxiliary input $z \in \{0,1\}^*$, and a polynomial $p(\cdot)$ such that for infinitely many values of $\kappa \in \mathbb{N}$ we have $p_z(\kappa) \geq 1/p(\kappa) + 2^{-\ell}$ in Definition 10, but for all PPT extractors $\mathcal{K}$

$$\Pr\left[(x,w) \in R : \begin{array}{l} (c,\vartheta) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa); \\ (x,a) \leftarrow\!\!\$\ \mathcal{P}^{*\mathcal{V}(\vartheta,\cdot,\cdot)}(c,z) \text{ where } |x| = \kappa; \\ w \leftarrow\!\!\$\ \mathcal{K}(1^\kappa, x, z, \mathcal{Q}). \end{array}\right] < 1/p(\kappa), \tag{8}$$

where $\mathcal{Q}$ is the list of verification queries (and answers to these queries) made by $\mathcal{P}^*$.

Consider the following adversary $\mathcal{A}$, running in the experiment $\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(b,\kappa)$:

- Upon input ek, run $(x,a) \leftarrow\!\!\$\ \mathcal{P}^*(ek, z)$.
- Upon input a query (x_i, a_i) from $\mathcal{P}^*$, forward x_i to the challenger receiving back a response y_i; return 1 to $\mathcal{A}$ iff $y_i = a_i$.
- Whenever $\mathcal{P}^*$ outputs (x,a), forward x receiving back a challenge y. In case $a = y$ output 1, else output 0.

We have:

$$\begin{aligned} \alpha_{\mathsf{wprf}}(R,\mathcal{A}) &= \left|\Pr[\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(0,\kappa) = 1] - \Pr[\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(1,\kappa) = 1]\right| \\ &= p_z(\kappa) - 2^{-\ell} \geq 1/p(\kappa), \end{aligned}$$

where the second equation comes from the fact that the answers to $\mathcal{A}$'s F-queries are uniformly random in the experiment $\mathbf{Exp}^{\mathsf{wprf}}_{R,\mathcal{A}}(1,\kappa)$, and by our assumption on the success probability of $\mathcal{P}^*$. Note that the above equation, together with Eq. (8), contradicts the fact that Π_{WPRF} is secure, as any PPT extractor would have a negligible advantage in extracting a valid witness for the value x returned by $\mathcal{A}$. The theorem follows. □

6.2 On Weak PZAP versus PZAP

We investigate the relation between the notions of weak PZAP and PZAP. On the positive side, in Section 6.2.1, we show that weak PZAP for *NP* can be generically leveraged to PZAP for *NP* in a generic (non-black-box) manner. On the negative side, in Section 6.2.2, we show an impossibility result ruling out a broad class of black-box reductions from weak PZAP to PZAP. Both results assume the existence of one-way functions.

6.2.1 From Weak PZAP to PZAP

We show the following result:

Theorem 10. *Under the assumption that non-interactive zero-knowledge proof of knowledge systems for NP and non-interactive computationally-hiding commitment schemes exist, weak PZAP for NP imply PZAP for NP.*

Before coming to the proof, let us introduce some useful notation. Given a set $I \subseteq \{0,1\}^\kappa$, we will say that I is *bit-fixing* if there exists a string in $x \in \{0,1,\star\}^\kappa$ such that $I_x = I$ where $I_x := \{y \in \{0,1\}^\kappa : \forall i \in [\kappa], (x_i = y_i \vee x_i = \star)\}$ is the set of all κ-bit strings matching x in the positions where x is equal to 0/1. The symbol $\star$ takes the role of a special "don't care" symbol. Notice that there is a bijection between the set $\{0,1,\star\}^\kappa$ and the family of all bit-fixing sets contained in $\{0,1\}^\kappa$; in particular, for any $I \subseteq \{0,1\}^\kappa$ there exists a unique $x \in \{0,1,\star\}$ such that $I = I_x$ (and viceversa). Therefore, in what follows, we use x and I_x interchangeably. We also enforce the empty set to be part of the family of all bit-fixing sets, by letting $I_\perp = \emptyset$ (corresponding to $x = \perp$).

We now give some intuition for the proof of Theorem 10 . The proof is divided in two main steps. In the first step, we define three algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. Roughly speaking, such a tuple constitutes a special type of signature scheme where the key generation algorithm Gen additionally takes as input a bit-fixing set$I$ and returns a secret key that allows to sign messages $m \notin I$. There are two main properties we need from such a signature scheme: (i) The verification key and any set of polynomially many (adaptively chosen) signature queries do not reveal any information on the set I; (ii) It should be hard to forge signatures on messages $m \in I$, even when given the set I and the secret key corresponding to I. A variation of such a primitive, with a few crucial differences, already appeared in the literature under the name of functional signatures [BGI14].[18] Fix now some NP-relation R. In the second step of the proof, we consider an augmented NP-relation where the witness of an instance (x, VK) is either a witness w for $(x, w) \in R$, or a valid signature of x under VK. We then construct a PZAP based on a weak PZAP and on a NIZK-PoK for the above augmented NP-relation.

The reduction from weak PZAP to PZAP uses a partitioning technique, similar to the one used to prove unforgeability of several signature schemes (see, e.g., [BB04, HK08, MTVY11, BSW13, FNV15a]). Intuitively, we can set the reduction in such a way that by sampling a random bit-fixing set I all the verification queries made by a succeeding prover for the PZAP will not be in I with good probability (and therefore such queries can be dealt with using knowledge of the signature key corresponding to I); this holds because the prover has no information on the set I, as ensured by property (i) defined above. On the other hand, the challenge x^* output by the prover will be contained in the set I, which will allow the reduction to break the weak PZAP. Here, is where we rely on property (ii) described above, so that the reduction is not able to forge a signature for x^*, and thus the extracted witness w^* must be a valid witness for $(x^*, w^*) \in R$.

Proof of Theorem 10. Let Com be a computationally hiding commitment scheme with message space $\{0,1,\star\}^\kappa \cup \{\perp\}$. Consider the following relation:

$$R_{\mathsf{com}} := \big\{(m, com), (x, r) : \quad com = \mathsf{Com}(x; r) \ \wedge \ m \notin I_x \ \big\}.$$

Let $\mathcal{NIZK} = (\ell, \mathsf{Prove}, \mathsf{Ver})$ be a NIZK-PoK for the relation R_{com}. We define the following tuple of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

- Algorithm Gen takes as input the security parameter and a string $x \in \{0,1,\star\}^\kappa \cup \{\perp\}$, samples $\omega \leftarrow\!\!\$\ \{0,1\}^{\ell(\kappa)}$, and defines $com := \mathsf{Com}(x; r)$ for some random tape r. It then outputs $VK := (\omega, com)$ and $SK := (\omega, x, r)$.
- Algorithm Sign takes as input a secret key SK and a message m, and outputs $\sigma := \pi \leftarrow\!\!\$\ \mathsf{Prove}(\omega, (m, com), (x, r))$.
- Algorithm Verify takes as input a verification key VK and a pair (m, σ), parses $VK := (\omega, com)$, and outputs the same as $\mathsf{Ver}(\omega, (m, com), \sigma)$.

[18] On a high level, the difference is that functional signatures allow to generate punctured signature keys, whereas our signature scheme allows to puncture the message space.

The lemmas below show two main properties of the above signature scheme.

Lemma 4. *For any PPT distinguisher $\mathcal{D}$, and any bit-fixing set $I \subseteq \{0,1\}^\kappa$, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that:*

$$\begin{aligned}\big|\Pr[\mathcal{D}^{\mathsf{Sign}(SK_I,\cdot)}(VK, I) :\ (VK, SK_I) \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa, I)]\\ - \Pr[\mathcal{D}^{\mathsf{Sign}(SK,\cdot)}(VK, I) :\ (VK, SK) \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa, \perp)]\big| \leqslant \nu(\kappa),\end{aligned}$$

where $\mathcal{D}$ is not allowed to query its oracle on messages $m \in I$.

Proof. We consider a series of hybrid experiments, where each hybrid is indexed by a bit-fixing set I and outputs the view of a distinghuisher $\mathcal{D}$ taking as input a verification key and the set I, while given oracle access to a signing oracle.

Hybrid $\mathcal{H}_1^I$: The first hybrid samples $(VK, SK) \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa, I)$ and runs the distinghuisher $\mathcal{D}$ upon input (VK, I) and with oracle access to $\mathsf{Sign}(SK, \cdot)$.

Hybrid $\mathcal{H}_2^I$: Let $\mathcal{Z}$ be the simulator of the underlying NIZK-PoK. The second hybrid samples $(\tilde{\omega}, \vartheta) \leftarrow\!\!\$\ \mathcal{Z}_0(1^\kappa)$ and defines $com := \mathsf{Com}(x; r)$ (for random tape r) and $\tilde{VK} = (\tilde{\omega}, com)$. It then runs the distinghuisher $\mathcal{D}$ upon input $(\tilde{VK}, I)$, and answers its oracle queries m by returning $\tilde{\sigma} \leftarrow\!\!\$\ \mathcal{Z}_1(\vartheta, (m, com))$.

The two claims below imply the statement of Lemma 4.

Claim 1. *For all bit-fixing sets I, we have $\{\mathcal{H}_1^I\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathcal{H}_2^I\}_{\kappa\in\mathbb{N}}$.*

Proof of claim. The only difference between the two experiments is in the way the verification key is computed and in how the signature queries are answered. In particular, the second experiment replaces the CRS with a simulated CRS and answers signature queries by running the ZK simulator of the NIZK. Note that the commitment com has the same distribution in both experiments.

Clearly, given any distinguisher that tells apart the two hybrids for some set I we can derive a distinguisher contradicting the unbounded zero-knowledge property of the NIZK. This concludes the proof. □

Claim 2. *Let $I_\perp := \emptyset$. For all bit-fixing sets I, we have $\{\mathcal{H}_2^I\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \{\mathcal{H}_2^{I_\perp}\}_{\kappa\in\mathbb{N}}$.*

Proof of claim. Given a PPT distinguisher $\mathcal{D}$ telling apart $\mathcal{H}_2^I$ and $\mathcal{H}_2^{I_\perp}$, we construct a PPT distinguisher $\mathcal{D}$ that breaks computational hiding of the commitment scheme. Distinguisher $\mathcal{D}'$ is given as input a value com' which is either a commitment to I or a commitment to $I_\perp$. Thus, $\mathcal{D}'$ simply emulates the view for $\mathcal{D}$ but uses com' instead of com.

The claim follows by observing that in case com' is a commitment to I the view generated by $\mathcal{D}'$ is identical to that in hybrid $\mathcal{H}_2^I$, whereas in case com' is a commitment to $I_\perp$ the view generated by $\mathcal{D}'$ is identical to that in hybrid $\mathcal{H}_2^{I_\perp}$. Hence, $\mathcal{D}'$ retains the same advantage as $\mathcal{D}$, a contradiction. □

□

Lemma 5. *For any PPT forger $\mathcal{F}$, and for any bit-fixing set I, there exists a negligible function $\nu : \mathbb{N} \to [0,1]$ such that the following holds:*

$$\Pr\left[m^* \in I \wedge \mathsf{Verify}(VK, m^*, \sigma^*) = 1 :\ \begin{array}{c}(m^*, \sigma^*) \leftarrow\!\!\$\ \mathcal{F}(I, r),\\ (VK, SK_I) := \mathsf{Gen}(1^\kappa, I; r)\end{array}\right] \leqslant \nu(\kappa).$$

Proof. We rely on the knowledge soundness property of the NIZK-PoK and on the binding property of the commitment scheme. By contradiction, assume that there exists a PPT forger $\mathcal{F}$, a bit-fixing set I_x, and some polynomial $p(\cdot)$, such that for infinitely many values of $\kappa \in \mathbb{N}$

$$\Pr\left[m^* \in I_x \wedge \mathsf{Ver}(\omega, (m^*, com), \sigma^*) = 1 : \begin{array}{c} r \leftarrow\!\!\$\, \{0,1\}^*, \omega \leftarrow\!\!\$\, \{0,1\}^\ell \\ com \leftarrow\!\!\$\, \mathsf{Com}(x; r) \\ (m^*, \sigma^*) \leftarrow\!\!\$\, \mathcal{F}(\omega, r, I_x) \end{array} \right] \geq 1/p(\kappa).$$

Consider the following adversary $\mathcal{B}$ attacking the binding property of the commitment scheme:

i) Upon input 1^κ, run $(\tilde{\omega}, \vartheta) \leftarrow\!\!\$\, \mathcal{K}_0(1^\kappa)$;
ii) Obtain $(m^*, \sigma^*) \leftarrow\!\!\$\, \mathcal{F}(\tilde{\omega}, r, I_x)$ for some $x \in \{0, 1, \star\}^\kappa$ and $r \leftarrow\!\!\$\, \{0,1\}^*$;
iii) Extract $(x', r') \leftarrow\!\!\$\, \mathcal{K}_1(\tilde{\omega}, \vartheta, (m^*, com), \sigma^*)$, where $com = \mathsf{Com}(x; r)$;
iv) Output $(x, r), (x', r')$ and m (as an auxiliary output).

By relying on the knowledge soundness property of the NIZK-PoK, and using the fact that the forger outputs an accepting proof with non-negligible probability, we obtain:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ wins}] &= \Pr\left[com = \mathsf{Com}(x'; r') \wedge (x, r) \neq (x', r') : \ ((x, r), (x', r')), m) \leftarrow\!\!\$\, \mathcal{B}(1^\kappa)\right] \\
&\geq \Pr\left[com = \mathsf{Com}(x'; r') \wedge m \notin I_{x'} \wedge m \in I_x : \ ((x, r), (x', r')), m) \leftarrow\!\!\$\, \mathcal{B}(1^\kappa)\right] - \nu(\kappa) \\
&\geq \Pr\left[m^* \in I_x \wedge \mathsf{Ver}(\omega, (m^*, com), \sigma^*) = 1 : \begin{array}{c} r \leftarrow\!\!\$\, \{0,1\}^*, \omega \leftarrow\!\!\$\, \{0,1\}^\ell \\ com \leftarrow\!\!\$\, \mathsf{Com}(x; r) \\ (m^*, \sigma^*) \leftarrow\!\!\$\, \mathcal{F}(\omega, r, I_x) \end{array} \right] \\
&\geqslant 1/p(\kappa) - \nu(\kappa),
\end{aligned}$$

for some negligible function $\nu(\cdot)$. The first inequality uses the fact that the condition $(m \notin I_{x'}) \wedge m \in I_x)$ implies $I_x \neq I_{x'}$ (and thus $x \neq x'$), and thus is sufficient for violating the binding property. This concludes the proof. □

We can now explain how to transform a weak PZAP for *NP* into a PZAP for *NP*. Let R be an *NP*-relation. Consider the following derived relation:

$$R' = \{((x, \mathit{VK}), w) : \ (x, w) \in R \ \vee \ \mathsf{Verify}(\mathit{VK}, x, w) = 1)\}.$$

Clearly, R' is in *NP*, so let $\Pi = (\mathsf{Chall}, \mathsf{Resp}, \mathsf{Predict})$ be a weak PZAP for R'. Define the following PZAP $\Pi' = (\mathsf{Chall}', \mathsf{Resp}', \mathsf{Predict}')$ for the relation R.

- Algorithm Chall' takes as input $(1^\kappa, x)$ and proceeds as follows:
  - Run $(c, \vartheta) \leftarrow\!\!\$\, \mathsf{Chall}(1^\kappa)$.
  - Sample $(\mathit{VK}, \mathit{SK}) \leftarrow\!\!\$\, \mathsf{Gen}(1^\kappa, \perp)$, and let the challenge be $c' := (c, \mathit{VK})$ and the trapdoor be $\vartheta' = (\vartheta, \mathit{VK})$.
- Algorithm Resp' takes as input $(1^\kappa, x, w, c')$, parses $c' := (c, \mathit{VK})$, and outputs $a := \mathsf{Resp}(1^\kappa, (x, \mathit{VK}), w, c)$.
- Algorithm $\mathsf{Predict}'$ takes as input $1^\kappa, \vartheta', x$, parses $\vartheta' := (\vartheta, \mathit{VK})$, and outputs $b := \mathsf{Predict}(\vartheta, (x, \mathit{VK}))$.

The lemma below concludes the proof of Theorem 10.

Lemma 6. *Let* Π *and* Π' *be as above. If* Π *is a weak PZAP for* R'*, then* Π' *is a PZAP for* R*.*

Proof. Given a prover $\mathcal{P}^*$ for Π', we construct a prover $\mathcal{P}_\alpha$ for Π' for a parameter $\alpha \in [\kappa]$ to be determined later. The description of $\mathcal{P}_\alpha$ follows.

- Upon input challenge c, choose $s \in \{0, 1, \star\}^\kappa$ in such a way that $\alpha := |\{i \in [\kappa] : s_i = \star\}|$. Sample $(\mathit{VK}, \mathit{SK}_I) \leftarrow\!\!\$\ \mathsf{Gen}(1^\kappa, I)$ for $I := I_s$, and forward the challenge $c' := (c, \mathit{VK})$ to $\mathcal{P}^*$.
- Upon input a verification query (x_i, a_i) from $\mathcal{P}^*$ behave as follows:
  - In case $x_i \in I$, stop simulating $\mathcal{P}^*$, pick a random $x^* \leftarrow\!\!\$\ \{0, 1\}^\kappa \setminus I$, and return the instance (x^*, VK) and answer $a^* := \mathsf{Resp}(1^\kappa, c, (x^*, \mathit{VK}), \mathsf{Sign}(\mathit{SK}_I, x^*))$.
  - In case $x_i \notin I$, compute $\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathit{SK}_I, x_i)$ and answer the verification query with 1 iff $a = \mathsf{Resp}(1^\kappa, c, (x, \mathit{VK}), \sigma)$.
- Whenever $\mathcal{P}^*$ outputs (x^*, a^*), if $x^* \in I$ output $((x^*, \mathit{VK}), a^*)$. Else pick a random $x^* \leftarrow\!\!\$\ \{0, 1\}^\kappa \setminus I$ and return the instance (x^*, VK) and answer $a^* := \mathsf{Resp}(1^\kappa, c, (x^*, \mathit{VK}), \mathsf{Sign}(\mathit{SK}_I, x^*))$.

We define the extractor for Π' (w.r.t. the relation R) to be the same as the extractor $\mathcal{K}$ for Π (w.r.t. the relation R'). It remains to bound the probability that $\mathcal{K}$ output a valid witness for the relation R.

Let *Good* be the event that $x^* \in I$ and all the x_i's corresponding to $\mathcal{P}^*$'s verification queries are such that $x_i \notin I$. Moreover, let Ext_R (resp. $\mathit{Ext}_{R'}$) be the event that $(x, w) \in R$ (resp. $((x, \mathit{VK}), w) \in R'$) where w comes from running the extractor $\mathcal{K}$ in the definition of PZAP. We can write:

$$\begin{aligned} \Pr[\mathit{Ext}_R] &\geqslant \Pr[\mathit{Ext}_R \wedge \mathit{Good}] && (9) \\ &\geqslant \Pr[\mathit{Ext}_{R'} \wedge \mathit{Good}] - \nu(\kappa) \\ &\geqslant \Pr[\mathit{Ext}_{R'}] - \Pr[\neg \mathit{Good}] - \nu(\kappa) \\ &\geqslant \big(\Pr[\mathcal{P}' \text{ succeeds}] - \nu'(\kappa)\big) - \Pr[\neg \mathit{Good}] - \nu(\kappa), && (10) \end{aligned}$$

for negligible functions $\nu(\cdot), \nu'(\cdot)$. Here, Eq. (9) holds because of Lemma 5, whereas Eq. (10) follows by knowledge soundness of Π.

Observe that, by definition of $\mathcal{P}_\alpha$, the success probability when we condition on the event *Good* not happening is overwhelming (this is because in that case $\mathcal{P}_\alpha$ just computes a valid signature, and thus it succeeds with overwhelming probability by completeness of Π), therefore:

$$\Pr[\mathcal{P}_\alpha \text{ succeeds}] \geqslant \Pr[\mathcal{P}_\alpha \text{ succeeds} | \mathit{Good}] \cdot \Pr[\mathit{Good}] + (1 - \nu''(\kappa)) \Pr[\neg \mathit{Good}],$$

for some negligible function $\nu''(\cdot)$. Combining the last two equations, we obtain that there exists a negligible function $\nu'''(\cdot)$ such that:

$$\Pr[\mathit{Ext}_R] \geqslant \Pr[\mathcal{P}_\alpha \text{ succeeds} | \mathit{Good}] \cdot \Pr[\mathit{Good}] - \nu'''(\kappa).$$

We analyse the probability that $\mathcal{P}_\alpha$ succeeds conditioning on *Good* and the probability of event *Good* separately. We claim that the first term is negligibly close to the success probability of $\mathcal{P}^*$. In fact, when the event *Good* happens, by Lemma 4, the view generated by $\mathcal{P}_\alpha$ is indistinguishable from the view in the knowledge soundness definition of PZAP.

As for the second term, again by Lemma 4, it is not hard to see that it is negligibly close to $(1 - 2^{-\kappa+\alpha})^Q \cdot 2^{-\kappa+\alpha}$, where Q is an upper bound for the number of verification queries made by the prover. Since when $2^{-\kappa+\alpha} := 1 - Q/(Q+1)$, then $(1 - 2^{-\kappa+\alpha})^Q \cdot 2^{-\kappa+\alpha} \geqslant 1/e$, it suffices to set $\alpha := \kappa + \log(1 - Q/(Q+1))$ to enforce that the probability of *Good* is noticeable. This concludes the proof. □

□

6.2.2 Ruling-Out Challenge-Passing Reductions

Below, we define what it means to reduce weak knowledge soundness to knowledge soundness of a PZAP Π (in a black-box way).

Definition 12. A PPT oracle machine $\mathcal{R}$ is called a black-box reduction from weak knowledge soundness to knowledge soundness of a PZAP Π if there exists a polynomial $q(\cdot)$ such that for any prover $\mathcal{P}^*$ making polynomially many verification queries, and any auxiliary input $z \in \{0,1\}^*$, the following holds. Let $p_z(\kappa)$ be the succeeding probability of $\mathcal{P}^*(\cdot, z)$; then $\mathcal{R}^{\mathcal{P}^*(\cdot,z)}(1^\kappa, \cdot)$ is a prover for Π that does not make any verification query and has success probability $q(p_z(\kappa))$.

Moreover, we say that $\mathcal{R}$ is *challenge-passing* if upon input a challenge c, the reduction simply forwards the same challenge c to $\mathcal{P}^*$.

The following theorem intuitively says that, if one-way functions exists, there cannot be a challenge-passing black-box reduction from weak knowledge soundness to knowledge soundness of a *laconic* PZAP.

Theorem 11. *Assume that one-way function exists, and let* Π *be a laconic weak PZAP for NP. There is no challenge-passing black-box reduction from weak knowledge soundness to knowledge soundness of* Π.

Proof. By contradiction, assume such a challenge-passing black-box reduction $\mathcal{R}$ exists. Consider the following prover $\mathcal{P}^*$ for the *NP*-relation $R_{\mathsf{prg}} := \{(G(s), s) : s \in \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$, where G is a PRG with one-bit stretch (i.e., $G(s) \in \{0,1\}^{\kappa+1}$).

1. **Check Verifier.** Compute $g_0 := G(s_0)$ for $s_0 \leftarrow\!\!\$\, \{0,1\}^\kappa$ and $a_0 \leftarrow\!\!\$\, \{0,1\}$. Forward (g_0, a_0) to the verification oracle, and let d_i be the corresponding answer from the oracle. Check whether $a_0 = \mathsf{Resp}(1^\kappa, c, (g_0, s_0))$ (i.e, the prover produced a valid predictable proof) if only if $d_i = 0$ (i.e., the verification oracle did not accept the proof). If that happens, abort and output $\perp$.
2. **Gain Information.** Let $p_\vartheta(\kappa)$ be a polynomial that upper bounds the length of the trapdoor ϑ produced by $\mathsf{Chall}(1^\kappa)$, and define $m(\kappa) := p_\vartheta(\kappa) + \kappa + 1$. Upon input challenge c, for all $i \in [m]$ sample $g_i \leftarrow\!\!\$\, \{0,1\}^{\kappa+1}$ and set $a_i \leftarrow\!\!\$\, \{0,1\}$. Forward (g_i, a_i) to the verification oracle, and let d_i be the corresponding answer from the oracle. If $d_i = 0$ set $a_i := 1 - a_i$.
(Notice that at the end of step 2 the answers $a_1, \ldots, a_m$ are all correct.)
3. **Sample Trapdoors.** Let $n = 6\kappa^2$. For all $i \in [n]$ sample $\tilde{\vartheta}_i$ at random from the distribution defined below:
$$\big\{\tilde{\vartheta} : \mathsf{Predict}(1^\kappa, \tilde{\vartheta}, g_i) = a_i, \forall i \in [m]\big\}.$$
4. **Compute Answer.** Sample $g^* \leftarrow\!\!\$\, \{0,1\}^{\kappa+1}$ and let $a^* := \mathsf{Maj}\{\mathsf{Predict}(1^\kappa, \tilde{\vartheta}_j, g^*) : j \in [n]\}$, and output (g^*, a^*).

Notice that prover $\mathcal{P}^*$ is unbounded. Nevertheless, we will later show that we can emulate $\mathcal{P}^*$ in polynomial time without any distinguisher being able to tell the difference. Before doing this, we prove that $\mathcal{P}^*$ succeeds with overwhelming probability.

Claim 3. *The probability that* $\mathcal{P}^*$ *succeeds is overwhelming.*

Proof of claim. Notice first that $\mathcal{P}^*$ aborts with probability bounded by the completeness error of Π, which is negligible. Therefore, for the rest of the proof, we condition on the event that $\mathcal{P}^*$ does not abort.

For any tuple $(\vartheta, \tilde{\vartheta}, g)$, define $Eq(\tilde{\vartheta}, \vartheta, g)$ to be the event that

$$\mathsf{Predict}(1^\kappa, \vartheta, g) = \mathsf{Predict}(1^\kappa, \tilde{\vartheta}, g).$$

Fix any ϑ. Given a trapdoor $\tilde{\vartheta}$, we say that $\tilde{\vartheta}$ is *Bad* if $\Pr[Eq(\tilde{\vartheta}, \vartheta, g) : \ g \leftarrow\!\!\$ \ \{0,1\}^{\kappa+1}] < 1/4$. It is easy to check that, for any $(\vartheta, \tilde{\vartheta})$, and for randomly sampled $g_1, \ldots, g_m \leftarrow\!\!\$ \ \{0,1\}^{\kappa+1}$:

$$\Pr\left[\bigwedge\nolimits_{i\in[m]} Eq(\tilde{\vartheta}, \vartheta, g_i) \mid \tilde{\vartheta} \text{ is } \mathit{Bad}\right] \leqslant (1 - 1/4)^m \leqslant 2^{-p_\vartheta(\kappa)-\kappa}. \tag{11}$$

We can now compute the following probability, for any fixed ϑ and for a randomly sampled $\tilde{\vartheta} \leftarrow\!\!\$ \ \{0,1\}^{p_\vartheta(\kappa)}$:

$$\begin{aligned}\Pr\left[\tilde{\vartheta} \text{ is } \mathit{Bad} \mid \bigwedge\nolimits_{i\in[m]} Eq(\tilde{\vartheta}, \vartheta, g_i)\right] &= \frac{\Pr\left[\bigwedge_{i\in[m]} Eq(\tilde{\vartheta}, \vartheta, g_i) \mid \tilde{\vartheta} \text{ is } \mathit{Bad}\right]}{\Pr\left[\bigwedge_{i\in[m]} Eq(\tilde{\vartheta}, \vartheta, g_i)\right]} \\ &\leqslant \Pr\left[\bigwedge\nolimits_{i\in[m]} Eq(\tilde{\vartheta}, \vartheta, g_i) \mid \tilde{\vartheta} \text{ is } \mathit{Bad}\right] \cdot 2^{p_\vartheta(\kappa)} \leqslant 2^{-\kappa}.\end{aligned}$$

Where the first equation follows by Bayes' rule for conditional probabilities, the second inequality follows because when $\tilde{\vartheta} = \vartheta$ the event at the denominator always happens (and therefore the probability of such an event is at least $2^{-p_\vartheta(\kappa)}$), and the last inequality follows by Eq. (11). Let *Good* be the event that $\{\forall i \in [m] : \tilde{\vartheta}_i$ is not *Bad*$\}$. By a union bound, we have that $\Pr[\mathit{Good} | \wedge_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)] \geqslant (1 - n \cdot 2^{-\kappa})$. Hence,

$$\begin{aligned}\Pr\left[\mathcal{P}^* \text{ succeeds}\right] &= \Pr\left[\mathcal{P}^* \text{ succeeds} \middle| \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \\ &\geqslant \Pr\left[\mathcal{P}^* \text{ succeeds} \middle| \mathit{Good}, \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \cdot \Pr\left[\mathit{Good} \mid \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \\ &\geqslant \Pr\left[\mathcal{P}^* \text{ succeeds} \middle| \mathit{Good}, \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \cdot (1 - n \cdot 2^{-\kappa}) \end{aligned} \tag{12}$$

Notice that the random variables $Z_j := Eq(\tilde{\vartheta}_j, \vartheta, g)$, where $j \in [n]$, $g \leftarrow\!\!\$ \ \{0,1\}^{\kappa+1}$ and $\tilde{\vartheta}_j \leftarrow\!\!\$ \ \{0,1\}^{p_\vartheta(\kappa)}$, are independent even conditioned on the event *Good*. In particular, for any $j \in [n]$, we have $\Pr[Z_j | \mathit{Good}] \geqslant 2/3$. Thus, by a Chernoff bound:

$$\Pr\left[\sum_{j\in[n]} Z_j > \frac{2n}{3} - \kappa \mid \mathit{Good}\right] \leqslant 2^{-O(\kappa)}.$$

Finally, we note that whenever $\sum_{j\in[n]} Z_j > \frac{n}{2}$ the prover $\mathcal{P}^*$ succeeds (as in such a case the answer is predictable). Moreover $\frac{2n}{3} - \kappa > \frac{n}{2}$, and therefore:

$$\begin{aligned}&\Pr\left[\mathcal{P}^* \text{ succeeds} \mid \mathit{Good}, \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \\ &\quad \geqslant \Pr\left[\sum_{j\in[n]} Z_j > \tfrac{n}{2} \mid \mathit{Good}, \bigwedge\nolimits_{i\in[m],j\in[n]} Eq(\tilde{\vartheta}_j, \vartheta, g_i)\right] \geqslant 1 - 2^{-O(\kappa)}.\end{aligned}$$

Combining the above equation with Eq. (12) yields the claim. □

We now turn to define the simulated prover $\tilde{\mathcal{P}}$. We give two alternative simulation strategies, according to two different cases.

Case 1: The transcript of the interaction between $\mathcal{R}$ and $\mathcal{P}^*$ contains the special symbol $\perp$ with overwhelming probability. In this case the simulated prover $\tilde{\mathcal{P}}$ simply runs the first step of $\mathcal{P}$, and then aborts.

Case 2: The transcript of the interaction between $\mathcal{R}$ and $\mathcal{P}^*$ does not contain the special symbol $\perp$ with noticeable probability. Specifically, let $p(\kappa)$ be a polynomial such that the probability that the transcript of the interaction between $\mathcal{R}$ and $\mathcal{P}^*$ does not contain $\perp$ is upper bounded by $1 - 1/p(\kappa)$. A description of $\tilde{\mathcal{P}}$ in such a case follows:

1. Upon input c, run $n := 2p(\kappa)^2 + \kappa$ different internal instances of $\mathcal{R}$ (let us call these instances $\tilde{\mathcal{R}}_1, \ldots, \tilde{\mathcal{R}}_n$) on input c. Since $\tilde{\mathcal{R}}_1, \ldots, \tilde{\mathcal{R}}_n$ are challenge-passing they will first return back c, and wait to receive the first query. Proceed in the same way as in the first step in the description of $\mathcal{P}^*$.
2. Proceed in the same way as in the second step in the description of $\mathcal{P}^*$.
3. Pick random $\tilde{g} \leftarrow\!\!\$ \{0,1\}^{\kappa+1}$, and for all $i \in [n]$ pick $\tilde{a}_i \leftarrow\!\!\$ \{0,1\}$ and forward $(\tilde{g}, \tilde{a}_i)$ to $\tilde{\mathcal{R}}_i$ which will return a decision bit $\tilde{d}_i$; in case $\tilde{d}_i = 0$ set $\tilde{a}_i^* := 1 - \tilde{a}_i$ otherwise set $\tilde{a}_i^* := \tilde{a}_i$.
4. Output $(\tilde{g}, \tilde{a})$ where $\tilde{a} := \mathsf{Maj}\{\tilde{a}_1^*, \ldots, \tilde{a}_n^*\}$.

(Notice that the polynomial $p(\kappa)$ depends only on the implementation of $\mathcal{R}$ and therefore can be passed to $\tilde{\mathcal{P}}$ via the auxiliary input.)

Claim 4. *The views produced by running $\mathcal{P}^*$ and $\tilde{\mathcal{P}}$ are computationally indistinguishable.*

Proof of claim. Notice that $\mathcal{P}^*$ and $\tilde{\mathcal{P}}$ proceed identically in the first two steps. In the third step, $\mathcal{P}^*$ outputs a uniformly chosen element g^* and a valid predictable proof a^*, while $\tilde{\mathcal{P}}$ outputs a uniformly chosen element $\tilde{g}$ and a predictable proof $\tilde{a}$ which was previously queried to $\tilde{\mathcal{R}}$.

We will show that since with noticeable probability the symbol $\perp$ is not contained in the transcript, and by indistinguishability of the PRG G, there exists a index i^* for which the proof $\tilde{a}$ is accepting for the challenge c and element $\tilde{g}$ with overwhelming probability. Therefore the simulated prover succeeds with overwhelming probability, and so the two views are indistinguishable.

Recall that the special symbol $\perp$ is contained in the transcript if $a_0 = \mathsf{Resp}(1^\kappa, c, (g_0, s_0))$ and $d_0 = 0$ or $a_0 \neq \mathsf{Resp}(1^\kappa, c, (g_0, s_0))$ and $d_0 = 1$ (i.e. $a_0 = \mathsf{Resp}(1^\kappa, c, (g_0, s_0)) \iff d_0 = 0$).

First, we note that by indistinguishability of the PRG G for any $i \in [n]$ the transcript of $\tilde{\mathcal{P}}$ with $\tilde{\mathcal{R}}_i$ would lead to $\perp$ with probability at most $1 - 1/p(\kappa) - negl(\kappa)$. Therefore, let Z_i be the random variable equal to 1 if $\tilde{a}_i^*$ is the correct answer for the challenge c and element $\tilde{g}$, we have that for any $i \in [n]$, $\Pr[Z_i = 1] \geqslant 1/2p(\kappa)$. Moreover, the random variables $Z_1, \ldots, Z_n$ are independent, therefore by a Chernoff bound and because the answer is predictable, we have that $\tilde{a}$ is correct with overwhelming probability. □

Claim 3 and Claim 4, and the definition of challenge-passing reduction, imply that the probability that $\mathcal{R}^{\tilde{\mathcal{P}}}$ succeeds is noticeable. Notice that if $\mathcal{R}$ exists then $\tilde{\mathcal{P}}$ can be efficiently implemented, and therefore $\mathcal{R}^{\tilde{\mathcal{P}}}$ is also efficient and it succeeds with noticeable probability in breaking knowledge soundness of the weak PZAP. This concludes the proof. □

7 Conclusion and Open Problems

We initiated the study of Predictable Arguments of Knowledge (PAoK) systems for NP. Our work encompasses a full characterization of PAoK (showing in particular that they can without loss of generality assumed to be extremely laconic), provides several constructions of PAoK (highlighting that PAoK are intimately connected to witness encryption and program obfuscation), and studies PAoK with additional properties (such as zero-knowledge and Predictable ZAP).

Although, the notions of PAoK and Ext-WE are equivalent, we think that they give two different points of view on the same object. Ultimately, this can only give more insights.

There are several interesting questions left open by our work. First, one could try to see whether there are other ways (beyond the ones we explored in the paper) how to circumvent the implausibility result of [GGHW14]. For instance it remains open if full-fledged PAoK for *NP* exist in the random oracle model.

Second, while it is impossible to have PAoK that additionally satisfy the zero-knowledge property in the plain model—in fact, we were able to achieve zero-knowledge in the CRS model and in the non-programmable random oracle model)—such a negative result does not apply to witness indistinguishability. Hence, it would be interesting to construct PAoK that are additionally witness indistinguishable in the plain model. An analogous question holds for PZAP.

Third, we believe the relationship between the notions of weak PZAP (where the prover is not allowed any verification query) and PZAP deserves further study. Our impossibility result for basing PZAP on weak PZAP in a black-box way, in fact, only rules out very basic types of reductions (black-box, and challenge-passing), and additionally only works for laconic PZAP. It remains open whether the impossibility proof can be extended to rule-out larger classes of reductions for non-laconic PZAP, or if the impossibility can somehow be circumvented using non-black-box techniques.

Finally, it would be interesting to find more applications for PAoK and PZAP (beyond the ones we mentioned in the introduction).

References

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013:689, 2013.

[AFP15] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline witness encryption. *IACR Cryptology ePrint Archive*, 2015:838, 2015.

[AL83] Dana Angluin and David Lichtenstein. Provable security of cryptosystems: A survey. Technical Report TR-288, Yale University, October 1983.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[BBC$^+$13] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for sphfs and efficient one-round PAKE protocols. In *CRYPTO*, pages 449–475, 2013.

[BCH12] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC*, pages 266–284, 2012.

[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.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519, 2014.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS*, pages 374–383, 1997.

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In *ASIACRYPT*, pages 102–121, 2014.

[BSW13] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. *J. Cryptology*, 26(3):513–558, 2013.

[BSW16] Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In *EUROCRYPT, Part II*, pages 792–821, 2016.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *TCC*, pages 17–33, 2005.

[CL10] Kai-Min Chung and Feng-Hao Liu. Parallel repetition theorems for interactive arguments. In *TCC*, pages 19–36, 2010.

[CP15] Kai-Min Chung and Rafael Pass. Tight parallel repetition theorems for public-coin arguments using KL-divergence. In *TCC*, pages 229–246, 2015.

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

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *Innovations in Computer Science*, pages 434–452, 2010.

[DS15] David Derler and Daniel Slamanig. Practical witness encryption for algebraic languages and how to reply an unknown whistleblower. *IACR Cryptology ePrint Archive*, 2015:1073, 2015.

[FMNV15] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von Neumann architecture. In *PKC*, pages 579–603, 2015.

[FN15] Antonio Faonio and Jesper Buus Nielsen. Fully leakage-resilient codes. *IACR Cryptology ePrint Archive*, 2015:1151, 2015.

[FNV15a] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Mind your coins: Fully leakage-resilient signatures with graceful degradation. In *ICALP*, pages 456–468, 2015.

[FNV15b] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. Cryptology ePrint Archive, Report 2015/740, 2015. `http://eprint.iacr.org/2015/740`.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, pages 518–535, 2014.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In *CRYPTO*, pages 536–553, 2013.

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

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

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[GOVW12] Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In *TCC*, pages 494–511, 2012.

[GVW02] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[Hai13] Iftach Haitner. A parallel repetition theorem for any interactive argument. *SIAM J. Comput.*, 42(6):2487–2501, 2013.

[HK08] Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In *CRYPTO*, pages 21–38, 2008.

[HPWP10] Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In *TCC*, pages 1–18, 2010.

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106, 2011.

[NVZ13] Jesper Buus Nielsen, Daniele Venturi, and Angela Zottarel. On the connection between leakage tolerance and adaptive security. In *PKC*, pages 497–515, 2013.

[OO89] Tatsuaki Okamoto and Kazuo Ohta. Divertible zero knowledge interactive proofs and commutative random self-reducibility. In *EUROCRYPT*, pages 134–148, 1989.

[PV07] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. An efficient parallel repetition theorem for Arthur-Merlin games. In *STOC*, pages 420–429, 2007.

[PW12] Krzysztof Pietrzak and Douglas Wikström. Parallel repetition of computationally sound protocols revisited. *J. Cryptology*, 25(1):116–135, 2012.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.

[SP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS*, pages 427–436, 1992.

[TW87] Martin Tompa and Heather Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *FOCS*, pages 472–482, 1987.

[Wee09] Hoeteck Wee. Zero knowledge in the random oracle model, revisited. In *ASIACRYPT*, pages 417–434, 2009.

[Wee10] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *CRYPTO*, pages 314–332, 2010.

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In *TCC*, pages 421–448, 2016.

A Application to Leakage-Tolerant Secure Message Transmission

In this section we show an application of PAoK. We show that if you use a public-key encryption scheme to implement secure message transmission and the resulting protocol can tolerate leakage of even a constant number of bits, then the scheme will have to have secret keys which are as long as the total number of bits transmitted using that key. Since the typical interpretation of public-key encryption is that an unbounded number of messages can be encrypted using a fixed public key, this shows that typical public-key encryption cannot be used to realize leakage-tolerant secure message transmission even against a constant number of leaked bits.

This strengthens an earlier result which showed this was the case for schemes tolerating a super-logarithmic number of bits of leakage. For the proof we need that there exists a PAoK for a relation associated to the public-key encryption scheme, essentially proving knowledge of the secret key. The result can therefore be interpreted as showing that either a public-key encryption does not give secure message transmission secure against a constant number of bits of leakage or there does not exists a PAoK for the applied public-key encryption scheme. Proving that such a PAoK does not exists seems very challenging using current techniques, so the result indicates that we probably cannot base leakage-tolerant message transmission secure against leaking a constant number of bits on public-key encryption with our current proof techniques.

Syntax of PKE. A tuple of algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a PKE scheme for message space $\mathcal{M}$ if the following holds: (i) Algorithm KGen takes as input the security parameter and returns a pair (pk, sk); (ii) Algorithm Enc takes as input a public key pk and a message $m \in \mathcal{M}$, and outputs a ciphertext γ; (iii) Algorithm Dec takes as input a secret key sk and a ciphertext γ, and outputs a message $m \in \mathcal{M}$ or $\perp$. We require that for all $m \in \mathcal{M}$ one has $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$ with overwhelming probability over the randomness of $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$.

Message transmission with leakage. We recall the notion of leakage tolerance—introduced by Bitansky *et al.* [BCH12]—for secure message transmission. Informally, in a leakage-tolerant message transmission protocol leakage queries from an adversary $\mathcal{A}$ are viewed as a form of partial corruptions, where $\mathcal{A}$ does not receive the complete state of the chosen party but just some function of it. Security is then achieved if such an adversary can be simulated in the UC framework [Can01]. Without loss of generality we will consider only dummy adversaries—adversaries which just carry out the commands of the environment. I.e., it is the environment which specifies all leakage queries. We will therefore completely drop the adversary in the notation for clarity.

Let Π_{PKE} be a protocol between a sender Sen and a receiver Rec. The ideal-world functionality for secure message transmission with leakage is depicted in Fig. 1. We say that Π_{PKE} is a leakage-tolerant secure implementation of $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ if there exists a simulator S such that no environment Z can distinguish between the real life protocol Π_{PKE} and S interacting with the

Functionality $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$

Running with parties $\mathsf{Rec}, \mathsf{Sen}$ and adversary S, the functionality $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$ is parametrized by the security parameter κ, message space $\mathcal{M}$ and the set of all admissible leakage functions Φ. Hence, $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$ behaves as follows:

- Upon input $(\mathtt{send}, \mathsf{Sen}, \mathsf{Rec}, m)$ send a message $(\mathtt{send}, \mathsf{Sen}, \mathsf{Rec}, |m|)$ to S. Once S allows to forward the message, send $(\mathtt{sent}, \mathsf{Sen}, m)$ to Rec.
- Upon input $(\mathtt{leak}, X, \phi_{\mathsf{Z}})$ for $X \in \{\mathsf{Sen}, \mathsf{Rec}\}$ and $\phi_{\mathsf{Z}} \in \Phi$ send a message $(\mathtt{leak}, X)$ to S. Receive $(\mathtt{leak}, X', \phi_{\mathsf{S}})$ from S, check that $\phi_{\mathsf{S}} \in \Phi$, and that $|\phi_{\mathsf{Z}}(\cdot)| = |\phi_{\mathsf{S}}(\cdot)|$ and $X' = X$. Send $(\mathtt{leak}, \phi_{\mathsf{S}}(m))$ to S and $(\mathtt{leaked}, |\phi_{\mathsf{S}}(m)|)$ to X'.

Figure 1: Ideal functionality $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$ for secure message transmission with leakage

ideal functionality $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$. We denote with $\mathbf{IDEAL}_{\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}},\mathsf{S},\mathsf{Z}}(\Phi, \kappa)$ the output of the environment Z when interacting with simulator S in the simulation.

Consider the following protocol Π_{PKE} between a sender Sen and a receiver Rec, supposed to realize $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$ via a public-key encryption scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$, assuming authenticated channels:

1. Sen transmits to Rec that it wants to forward a message $m \in \mathcal{M}$;
2. Rec samples $(pk, sk) = \mathsf{KGen}(1^\kappa; r_G)$, and sends pk to Sen;
3. Sen computes $\gamma = \mathsf{Enc}(pk, m; r_E)$ and forwards the result to Rec;
4. Rec outputs $m' = \mathsf{Dec}(sk, \gamma; r_D)$.

Note that at the end of the execution of Π_{PKE} the state of Sen is $\sigma_S = (m, r_E)$ whereas the state of Rec is $\sigma_R = (sk, r_G, r_D, m')$. Denote with $\mathbf{REAL}_{\Pi_{\mathsf{PKE}},\mathsf{Z}}(\Phi, \kappa)$ the output of the environment Z after interacting with parties $\mathsf{Rec}, \mathsf{Sen}$ in a real execution of Π_{PKE}.

Definition 13 (Leakage-tolerant PKE protocol)**.** We say that Π_{PKE} is a (λ, ϵ)-leakage-tolerant PKE protocol (w.r.t. a set of leakage functions Φ) if Π_{PKE} securely implements $\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}}$, i.e., there exists a probabilistic polynomial-time simulator S such that for any environment Z leaking at most λ bits of information it holds that

$$\{\mathbf{IDEAL}_{\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}},\mathsf{S},\mathsf{Z}}(\Phi, \kappa)\}_{\kappa \in \mathbb{N}} \approx_\epsilon \{\mathbf{REAL}_{\Pi_{\mathsf{PKE}},\mathsf{Z}}(\Phi, \kappa)\}_{\kappa \in \mathbb{N}}.$$

Necessity of long keys. Nielsen *et al.* [NVZ13] have shown that any leakage-tolerant PKE protocol (as per Definition 13) requires long keys already when Π_{PKE} tolerates super-logarithmic leakage. Below we strengthen their result proving a more fine-grained lower bound for any $\lambda = O(1)$ bits of leakage.

Consider the following *NP* relation, depending upon a PKE scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$:

$$R_{\mathsf{PKE}} := \{((pk, \gamma, m), (sk, r_G)) : \ (pk, sk) = \mathsf{KGen}(1^\kappa; r_G) \wedge \mathsf{Dec}(sk, \gamma) = m\} \qquad (13)$$

We show the following theorem.

Theorem 12. *Let* $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ *be a PAoK for the above relation* R_{PKE}*, with knowledge soundness error* $2^{-\ell} + \epsilon$*, perfect completeness, and prover's answers of length* $\ell \geq 1$*. Let* Π_{PKE} *be an* (ℓ, ϵ')*-leakage-tolerant PKE protocol. Then it must be that* $|\mathcal{SK}| \geqslant (1 - 2^{-\ell} - \epsilon - \mu - 2\epsilon')|\mathcal{M}|$*, where* $\mathcal{SK}$ *is the space of all secret keys.*

Before we sketch the proof, we give an interpretation of the Theorem 12. When the knowledge soundness of the PAoK is negligible and the PKE protocol is $(\ell, negl)$-leakage-tolerant it means that $|\mathcal{SK}| \geqslant (1 - 2^{-\ell+1})|\mathcal{M}|$

Proof sketch. We make the proof for the case where the decryption algorithm is deterministic and has perfect correctness. Probabilistic decryption can be handled as in [NVZ13].

We construct an environment Z which uses ℓ bits of leakage on the receiver's state after the execution of Π, for which the existence of a simulator S implies our bound. The environment Z works as follows:

1. Input a uniformly random $m \in \mathcal{M}$ to Sen.
2. Let the protocol terminate without any leakage queries or any corruptions, i.e., simply deliver all messages between Sen and Rec. As part of this Z learns pk and γ from observing the authenticated channel between Sen and Rec.
3. After the protocol terminates, let Rec prove via leakage queries that $x := (pk, c, m) \in L_{R_{\mathsf{PKE}}}$ using $\Pi = (\mathsf{Chall}, \mathsf{Resp})$. Notice that Rec can do this as it knows a valid witness $w := (sk, r_G)$. More in detail, Z runs $(c, b) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa, x)$ and specifies a single leakage query defined as $\phi_{\mathsf{Z}}^{x,c}(\sigma_{\mathsf{Rec}}) = \phi_{\mathsf{Z}}^{x,c}(w) = \mathsf{Resp}(1^\kappa, x, w, c)$ (the values x and c are hard-wired into the function).
4. Finally Z outputs 1 if and only if $a = b$, where a is the output of Z's leakage query.

Note that the total amount of leaked information is equal to the communication complexity of the prover in Π, i.e., ℓ bits. By completeness of the PAoK, we know that $\mathbf{REAL}_{\Pi_{\mathsf{PKE}},\mathsf{Z}}(\Phi, \kappa) = 1$. From this we conclude that $\mathbf{IDEAL}_{\mathbf{F}^{+\mathsf{lk}}_{\mathsf{SMT}},\mathsf{S},\mathsf{Z}}(\Phi, \kappa) = 1$ except with negligible probability, by security of the protocol. We write out what this means. The simulation proceeds as follows:

1. First Z inputs a uniformly random $m \in \mathcal{M}$ to the ideal functionality on behalf of Sen. As a result S is given $(\mathtt{send}, \mathsf{Sen}, \mathsf{Rec}, |m|)$.
2. Then S must simulate the communication of the protocol, which in particular means that it must output some pk and γ to Z.
3. After the simulation of the protocol terminates, the environment runs $(c, b) \leftarrow\!\!\$\ \mathsf{Chall}(1^\kappa, x)$ and makes the leakage query $\phi_{\mathsf{Z}}^{x,c}$ with which Rec proves that $x = (pk, \gamma, m) \in L_{R_{\mathsf{PKE}}}$. Such leakage query is answered by S using another function ϕ_{S} producing a value a.
4. Finally Z outputs 1 if and only if $a = b$

Since $\mathbb{Z}$ is computing (c, b) as the verifier of Π would have done, and the value a is computed by S which is PPT, and since Z outputs 1, it follows from soundness that $x \in L_{R_{\mathsf{PKE}}}$ except with probability $\epsilon = 2^{-\ell} + negl(\kappa)$. This means that there exist (sk, r_G) such that $(pk, sk) = \mathsf{KGen}(1^\kappa; r_G)$ and $m = \mathsf{Dec}(sk, \gamma)$. In particular, there exists $sk \in \mathcal{SK}$ such that $m = \mathsf{Dec}(sk, \gamma)$. Let $M_{pk,\gamma} \subset \mathcal{M}$ denote the subset of $m' \in \mathcal{M}$ for which there exist $sk' \in \mathcal{SK}$ such that $m' = \mathsf{Dec}(sk', \gamma)$. An argument identical to the one in [NVZ13, Theorem 1] shows that $m \in M_{pk,\gamma}$ except with probability $\epsilon + \epsilon'$. Combined with the above, this implies that $|M_{pk,\gamma}| \geq (1 - 2^{-\ell} - negl(\kappa) - 2\epsilon')|\mathcal{M}|$. Take two $m_0 \neq m_1 \in M_{pk,\gamma}$. By definition there exist $sk_0, sk_1 \in \mathcal{SK}$ such that $m_0 = \mathsf{Dec}(sk_0, \gamma)$ and $m_1 = \mathsf{Dec}(sk_1, \gamma)$. From $m_0 \neq m_1$, we conclude that $sk_0 \neq sk_1$, so $|\mathcal{SK}| \geq |M_{pk,\gamma}|$. From this we get the theorem. □