Predictable Arguments of Knowledge

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

[1]*Department of Computer Science, Aarhus University* *
[2]*Department of Computer Science, Sapienza University of Rome* †

July 23, 2015

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 answers of the prover can be predicted, given the private randomness of the verifier.

We show that predictable arguments of knowledge (PAoK) can be made extremely laconic, with the prover sending a single bit, and assumed to have only one round (two messages) without loss of generality. We then explore constructs of PAoK. For specific relations we obtain PAoK from Extractable Hash Proof systems (Wee, Crypto '10); we also show that PAoK are equivalent to Extractable Witness Encryption. Unfortunately, the latter poses serious doubts on the existence of PAoK for all NP. However, we show that for the class of random self-reducible problems in NP we can avoid the problem relying on the assumption of public-coin differing-inputs obfuscation (Ishai *et al.*, TCC '15).

Finally, we apply PAoK in the context of leakage-tolerant PKE protocols. At PKC '13 Nielsen *et al.* have shown that any leakage-tolerant PKE protocol requires long keys already when it tolerates super-logarithmic leakage. We strengthen their result proving a more fine-grained lower bound for any constant numbers bits of leakage.

*Electronic Addresses: `{antfa,jbn}@cs.au.dk`
†Electronic Address: `venturi@di.uniroma1.it`

1 Introduction

Consider the classical proof system for Graphs Non-Isomorphism where, on common input a tuple of graphs (G_0, G_1), the verifier choses a uniformly 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 can reply 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 interesting 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.* in [14] we call a proof system with this property *extremely laconic*. We study the notion of predictability in interactive proof systems for NP, specifically, we focus on the more 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. We formalise this notion of Predictable Arguments of Knowledge (PAoK) and explore their properties and applications.

1.1 Our Contributions

Characterizing PAoK. In Section 3, we show that PAoK can always be made extremely laconic both in term of round complexity and in term of the number of bits send by the prover (i.e. message complexity). For the former we show that we can collapse any multi-round PAoK into a one-round PAoK with higher message complexity. For the latter we show how to reduce the message length to a single bit using the Goldreich-Levin Hard-Core Theorem (see Goldreich and Levin [13]). 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.

We sketch the main idea used in the first step. Consider a PAoK with round complexity ρ and a random sequence of challenges $(c_1, \ldots, c_\rho)$ where c_i is for round i. Without loss of generality the verifier can define all the challenge before any answers of the prover. Let $(a_1, \ldots, a_\rho)$ be the predicted answers. Consider for simplicity a cheating prover who can answer all of these challenges with probability $1/2$. Then for many possible challenges $(c_1, \ldots, c_\rho)$ there must exists a round i such that the prover cannot answer c_i with a_i with probability 1 even if it is given $(c_1, \ldots, c_{i-1})$ and $(a_1, \ldots, a_{i-1})$. Hence the verifier could sample $(c_1, \ldots, c_\rho)$ and $(a_1, \ldots, a_\rho)$, choose a uniformly random i, send $(c_1, \ldots, c_{i-1})$ and $(a_1, \ldots, a_{i-1})$ and expect to get back a_i. This would give non-zero probability of catching the cheating prover on many instances.

A main problem with this idea, however, is that if the round complexity of the protocol grows with the security parameter (which is the interesting case), the soundness will be vanishing. Therefore, we would like to amplify the soundness, however we cannot use sequential composition, as we are trying to construct a one-round protocol. Furthermore, in general, interactive arguments do not amplify soundness when compose in parallel (see Bellare *et al.* [2] and Pietrzak and Wikström [22]). It turns out that a more involved construction, inspired by parallel composition of the above simple protocol will actually allow to boost the soundness negligibly close to 1. We refer the reader to Section 3.2 for the details.

Constructions. In Section 4 we explore constructs of PAoK. In Section 4.1 we show how to obtain a PAoK, for some specific relations, using an Extractable Hash-Proof system (see Wee [24]).

Next, we show that PAoK for a relation R is equivalent to Extractable Witness Encryption (Ext-WE) (see Goldwasser *et al.* [15]) for R. The main ideas are the following. From Ext-WE to PAoK we encrypt a random bit a using the encryption scheme and then ask the prover to return a. For the other direction, namely from PAoK to Ext-WE, we first make the PAoK extremely laconic, generate a challenge/answer pair (c, a) for the PAoK, and then encrypt a single bit β as $(c, a \oplus \beta)$. We give the details in Section 4.2 .

The equivalence between PAoK and Ext-WE can be seen as a negative result for PAoK, as Ext-WE is implausible to exist for all of NP (see Garg *et al.* [9]). For this reason we relax the knowledge soundness requirement of both PAoK and Ext-WE to weaker variants where the extractor is given the randomness used to sample the instance; this approach is inspired by the notion of public-coin differing-inputs obfuscation (diO) (see Ishai *et al.* [18]). Weak PAoK and weak Ext-WE are still equivalent. We then show that weak Ext-WE can be constructed from public-coin diO. The main idea is to obfuscate, for a given instance x, the circuit which on input w such that (x, w) is in the relation outputs the message. As a corollary we get a weak PAoK for all NP. See Section 5 for the details.

We note that the lack of a (full) auxiliary input in the notion of weak PAoK means that it does not even have sequential composition, which makes the notion much harder to work with. We can in particular not prove that weak PAoK can always be made extremely laconic.

Finally, we show that if an NP relation R has a weak PAoK and R is random self reducible, then there also exists a PAoK for R. As a corollary we get PAoK for all random self-reducible languages in NP. Roughly speaking, a language is random self reducible if solving any instance is as hard as solve a random instance. Namely, there exists a reduction that allows to solve an instance given the solution for a random one. The main idea of the protocol is that the verifier "translates" the given instance x to a random instance x' and execute the weak PAoK over x' with the prover. Notice that the prover is an efficient machine, therefore even given the witness w for the x, it cannot solve the instance x'. To solve this we define the natural notion of witness reconstructible relation that allows to reconstruct the witness for x' given the randomness that generates x' from x and a valid witness w for x. The verifier therefore sends both the challenge for x' and also the randomness. Here is where the notion of weak PAoK kicks in, in fact, the protocol remains knowledge sound even when the prover knows the randomness used to create the instance. See Section 4.3 for the details.

Applications. We mainly investigate the notion of PAoK because we find it intriguing in its own right. We do, however, note that PAoK have a number of interesting properties and applications. Note for instance that a PAoK clearly is honest-verifier zero-knowledge, and therefore can be made zero-knowledge using standard reductions. In Appendix A we show an application of PAoK to proving lower bounds on the complexity of leakage-tolerant protocols. In particular, we show that if one uses a public-key encryption (PKE) scheme to implement secure message transmission, and the resulting protocol can tolerate leakage of even a constant number of bits, then the scheme needs to have secret keys which are as long as the total number of bits transmitted using that key.

Since the typical interpretation of PKE is that an unbounded number of messages can be encrypted using a fixed public key, this shows that typical PKE cannot be used to realize leakage-tolerant secure message transmission even against a constant number of leaked bits. This strengthens an earlier result by Nielsen *et al.* [19] who showed a lower bound for schemes tolerating super-logarithmic leakage.

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

1.2 Related Work

A study of interactive proofs with laconic provers was done also in [12, 14]. They do not investigate proofs of *knowledge*. As explained above our notion of PAoK is related to Ext-WE first proposed by Goldwasser *et al.* in [15], where they directly assume that the construction of Garg *et al.* in [10] is extractable. Our technique for weak PAoK from public-coin diO is related to the work of Boyle *et al.* [4].

In [9] Garg *et al.* showed that the notion of Ext-WE is "implausible", namely, assuming a virtual black-box obfuscation (VBB) for a specific function then Ext-WE for a specific NP relation is impossible. The reason relies on the auxiliary information that an adversary might have on the input, if such kind of VBB exists then the auxiliary input can be an obfuscated circuit that allows to decrypt ciphertexts but does not give any information about the witness. As stated by the authors of [9], this can be interpreted as an "implausibility" of the notion of Ext-WE (for all of NP). Notice that the result holds only when the adversary gets an arbitrary auxiliary input.

As mentioned above, in general, arguments do not compose nicely in parallel, however there are some exceptions like 3-messages arguments [2, 6], public-coin arguments [21, 8] and *simulatable* (a generalization of both 3-messages and public-coin) arguments [17, 7]. (multi-round) PAoK are inherently private coins so the mentioned works do not apply directly. More relevant to ours is the work of Haitner on random-terminating arguments [16].

2 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 A is an algorithm, we write $y \leftarrow \mathsf{A}(x)$ to denote a run of A on input x and output y; if A is randomized, then y is a random variable and $\mathsf{A}(x; r)$ denotes a run of A on input x and randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0, 1\}^*$ the computation of $\mathsf{A}(x; r)$ terminates in at most $\mathsf{poly}(|x|)$ steps.

Throughout the paper we let $\kappa \in \mathbb{N}$ denote the security parameter. We say that a function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible in the security parameter κ if $\mu(\kappa) = \kappa^{-\omega(1)}$. A positive function ν is noticeable if there exist a positive polynomial $p(\cdot)$ and a number κ_0 such that $\nu(\kappa) \geqslant 1/p(\kappa)$ for all $\kappa \geqslant \kappa_0$. For two ensembles $\mathcal{X} = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$, we write $\mathcal{X} \equiv \mathcal{Y}$ if they are identically distributed and $\mathcal{X} \approx \mathcal{Y}$ to denote that they are statistically or computationally close.

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 $\mathbf{v}$ and $\mathbf{v}_{\downarrow i}$ for the vector $(v_1, \ldots, v_i)$.

3 Predictable Arguments of Knowledge

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 for R features a prover P (holding a value $x \in L_R$ together with a corresponding witness w) and a verifier 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 $\texttt{acc}$ or $\texttt{rej}$; we write $\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle$ for the random variable corresponding to the verifier's verdict.

We start by defining Predictable Arguments of Knowledge (PAoK) in Section 3.1 as one-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 one-round and extremely laconic (i.e., the prover sends a single bit). In particular, 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 $\ell \in \mathbb{N}$, there exists a laconic PAoK if and only if there exists a PAoK where the prover's answer is of length ℓ.

3.1 The Definition

We focus on one-round protocols where the verifier speaks first by sending a challenge message c, to which the prover returns an answer a. Importantly, we are interested in protocols where the verifier can predict the prover's answer at the time when it generates the challenge. Such protocols are fully specified by a tuple of two ppt algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ as described below:

1. V samples $(c, b) \leftarrow \mathsf{Chall}(1^\kappa, x)$ and sends c to P.
2. P samples $a \leftarrow \mathsf{Resp}(1^\kappa, x, w, c)$ and sends a to V.
3. V outputs $\mathtt{acc}$ if and only if $a = b$.

We say that prover P and verifier 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 $\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle_\Pi$ (or simply $\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle$ when Π is clear from the context) the output of such interaction. We say that a prover P *succeeds* on the instance x and auxiliary input w if $\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle = \mathtt{acc}$.

Definition 1 (Predictable Arguments of Knowledge)**.** Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be as specified above, and let R be an NP relation. Consider the properties below.

Completeness: There exists a negligible function μ such that for all $(x, w) \in R$, we have that:

$$\Pr_{\mathsf{P},\mathsf{V}}\left[\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle = \mathtt{acc}\right] \geqslant 1 - \mu(\kappa).$$

f-Knowledge soundness with error ϵ: For all ppt provers P^* there exists a non-uniform extractor K and a non-zero polynomial $q(\cdot)$ such that for any $x \in \{0,1\}^*$ and any auxiliary input $z \in \{0,1\}^*$ the following holds. Whenever $p(\kappa) = \Pr[\langle \mathsf{P}^*(1^\kappa, x, z), \mathsf{V}(1^\kappa, x)\rangle = \mathtt{acc}] > \epsilon(\kappa)$, then

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

Let ℓ be the size of the prover's answer, we call Π a predictable argument of knowledge (PAoK) for R if Π satisfies completeness and f-knowledge soundness for any efficient computable function f, and moreover $\epsilon - 2^{-\ell}$ is negligible. We call it a *laconic* PAoK if $\ell = 1$. We call it an f-PAoK if knowledge soundness holds for a specific function f.

We now turn to define a weaker form of extractability, that roughly says that the protocol is sound and moreover random instances are extractable. To gain in generality we consider random elements from any efficiently samplable distribution. Consider the property below.

Weak f-Knowledge soundness with error ϵ: For all ppt provers P^* and all ppt algorithms Sam there exists a non-uniform extractor K and a non-zero polynomial $q(\cdot)$ such that for all auxiliary inputs $z_P, z_S \in \{0,1\}^*$ the following holds. Whenever $p(\kappa) := \Pr[\langle \mathsf{P}^*(1^\kappa, r, z_P), \mathsf{V}(\mathsf{Sam}(1^\kappa, z_S; r))\rangle = \mathtt{acc}] > \epsilon(\kappa)$, then

$$\Pr_{\mathsf{K},r}\left[\begin{array}{c}\exists w \text{ s.t. } f(w) = y\\ (x,w) \in R\end{array} : \begin{array}{c}x := \mathsf{Sam}(1^\kappa, z_s; r),\\ y \leftarrow\!\!\$\ \mathsf{K}(1^\kappa, r, z_p, z_s)\end{array}\right] \geqslant q(p(\kappa) - \epsilon(\kappa)).$$

Definition 2 (Weak PaoK)**.** Let $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be as specified above, and let R be an NP relation. We call Π a weak PAoK (wPAoK) for R if Π satisfies completeness and weak knowledge soundness with error ϵ, and moreover $\epsilon - 2^{-\ell}$ is negligible (where ℓ is the length of the prover's answer). We call it an f-wPAoK if weak knowledge soundness holds for a specific function f.

Notice that weak knowledge soundness connects the average-case hardness of a language (with instances sampled over a public-coin distribution) to the correctness of the argument system. However, given a prover that succeeds on a randomly sampled instance x in a weak PAoK the definition does not give any guarantee that the knowledge extractor is able to extract a witness for the same random instance x.

The above might seem to be a quite weak extraction guarantee. However, in Section 4.3 we will show that a weak PAoK can be used to obtain a full-fledged PAoK for languages that are random self-reducible.

3.2 On Multi-Round PAoK

In this section we consider a natural extension of predictable arguments where there are $\rho > 1$ rounds. In particular, we show that multi-round PAoK can be squeezed into a one-round PAoK (maintaining knowledge soundness).

In a multi-round predictable argument the verifier produces many challenges $\mathbf{c} = (c_1, \ldots, c_\rho)$. W.l.o.g. 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 PAoK is fully specified by a tuple of algorithms $\Pi = (\mathsf{Chall}, \mathsf{Resp})$, as described below:

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

Notice that now algorithm Resp takes as input all challenges up-to round i in order to generate the i-th answer.[1]

Definition 3 (ρ-round PAoK)**.** Let R be an NP relation, $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ be as above, and denote by ℓ the size of each answer produced by the prover. We call Π a ρ-round PAoK for R if Π satisfies completeness and knowledge soundness with error ϵ, and moreover $\epsilon - 2^{-\rho\ell}$ is negligible.

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

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

1. For all $i \in [\varphi]$ and $j \in [\rho]$, V' samples $(\mathbf{c}^{i,j}, \mathbf{b}^{i,j}) \leftarrow\$ \mathsf{Chall}(1^\kappa, x)$. For each $j \in [\rho]$ let:

$$\mathbf{C}^j := (\mathbf{c}^{1,j}_{\downarrow j}, \mathbf{c}^{2,j}_{\downarrow j}, \ldots, \mathbf{c}^{\varphi,j}_{\downarrow j})$$
$$\mathbf{B}^j := (\mathbf{b}^{1,j}_{\downarrow j}, \mathbf{b}^{2,j}_{\downarrow j}, \ldots, \mathbf{b}^{\varphi,j}_{\downarrow j}).$$

V' sends the vectors $\mathbf{C} = \left(\mathbf{C}^1, \mathbf{C}^2, \ldots, \mathbf{C}^\rho\right)$.

2. For all $i \in [\varphi]$ and $j \in [\rho]$, P' computes the answer $\mathbf{a}^{i,j} \leftarrow\$ \mathsf{Resp}(1^\kappa, x, w, \mathbf{C}^j[i])$. For each $j \in [\rho]$ let:

$$\mathbf{A}^j := (\mathbf{a}^{1,j}, \mathbf{a}^{2,j}, \ldots, \mathbf{a}^{\varphi,j}).$$

P' sends the vector $\mathbf{A} := (\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^\rho)$.

3. V' sets $\mathbf{B} := (\mathbf{B}^1, \mathbf{B}^2, \ldots, \mathbf{B}^\rho)$ and outputs $\mathtt{acc}$ if and only if $\mathbf{A} = \mathbf{B}$.

We write $\Pi' := (\mathsf{Resp}'_\varphi, \mathsf{Chall}'_\varphi)$ for the algorithms describing the generation of the challenge $\mathbf{C}$ and the predicted answer $\mathbf{B}$ by the prover, and the generation of the answer $\mathbf{A}$ by the verifier in the collapsed protocol (respectively).

Theorem 1. *For any polynomial $\rho(\cdot)$ and any function f if $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ is a $\rho(\kappa)$-round f-PAoK with knowledge error $\epsilon(\kappa)$, then the above collapsed protocol $\Pi' = (\mathsf{Chall}'_\varphi, \mathsf{Resp}'_\varphi)$ with parameter $\varphi = \omega(\log\kappa) \cdot \rho$ is an f-PAoK with knowledge error $\epsilon^{\omega(\log\kappa)}$.*

Proof. We start showing knowledge soundness of Π'. Let P' be a prover for the collapsed protocol, and define $p' := \Pr[\langle \mathsf{P}'(1^\kappa, x, z), \mathsf{V}'(1^\kappa, x)\rangle = \mathtt{acc}]$. We construct a prover P^* (with oracle access to P') that convinces the verifier V of the ρ-round PAoK Π with probability greater than $1/3$.

We start by setting up some notation. For any vector $\mathbf{c}$, any challenge $\mathbf{C}$ as defined in the collapsed protocol, and any pair of indexes i, j, let $\mathbf{C}_{|\mathbf{c}^{i,j}=\mathbf{c}}$ be defined as $\mathbf{C}$ but with the vector $\mathbf{c}^{i,j}$ set to $\mathbf{c}$. Let $Z_i(\mathbf{B}, \mathbf{C})$ be the indicator random variable for the event

$$\forall j \in [\rho]: \ \mathbf{a}^{i,j} = \mathbf{b}^{i,j}, \text{where } \mathbf{A} = \mathsf{P}'(1^\kappa, \mathbf{C}).$$

For any $i \in [\varphi]$ define δ_i to be the following probability:

$$\delta_i := \Pr\left[\sum_{k=1}^{\varphi} Z_k(\mathbf{B}, \mathbf{C}) = i : \ (\mathbf{B}, \mathbf{C}) \leftarrow\$ \mathsf{Chall}'_\varphi(1^\kappa, x)\right].$$

Finally, let $k^* \in [0, \varphi - 1]$ be the first index for which the following holds:

$$\frac{k^*+1}{\varphi}\delta_{\varphi-k^*-1} \leqslant \frac{1}{2\rho}\sum_{i \geqslant \varphi-k^*}\delta_i. \tag{1}$$

Before proceeding with the proof, we establish that for any P' that succeeds with noticeable probability p' there exists an index k^* having the property required in Eq. (1). This can be seen as follows. Suppose that Eq. (1) does not hold then:

$$\begin{aligned}\delta_1 &> \frac{\varphi}{2\rho}\sum_{i=2}^{\varphi}\delta_i = \frac{\varphi}{2\rho}\left(\delta_2 + \sum_{i=3}^{\varphi}\delta_i\right)\\ &> \frac{\varphi}{2\rho}\left(\left(\frac{\varphi}{2\cdot 2\rho}\sum_{i=3}^{\varphi}\delta_i\right) + \sum_{i=3}^{\varphi}\delta_i\right) = \frac{\varphi}{2\rho}\left(\frac{\varphi}{2\cdot 2\rho} + 1\right)\sum_{i=3}^{\varphi}\delta_i\\ &> \frac{\varphi}{2\rho}\cdot\prod_{i=2}^{\varphi-1}\left(1 + \frac{\varphi}{i2\rho}\right)\cdot p' > \frac{\varphi}{2\rho}\cdot\left(1 + \frac{1}{2\rho}\right)^{\varphi-3}\cdot p' > 1,\end{aligned}$$

where the last inequality holds if we set $\varphi = \omega(\log\kappa)\rho$. We reached a contradiction.

For any $i \in [\varphi]$, and for a parameter $\tau \in \mathbb{N}$, we define the procedure $\overline{\mathsf{Resp}}_{\tau}^{\mathsf{P}'}(1^{\kappa}, x, c_1, \ldots, c_j)$ that upon input an instance x and challenges $\mathbf{c} = (c_1, \ldots, c_j)$, and given oracle access to a prover P', outputs answers $a_1, \ldots, a_j$. The description of the procedure follows:

1. For τ many trials:
   - Sample $(\mathbf{C}, \mathbf{B})$ as in the collapsed protocol and choose $i^* \leftarrow\$ [\varphi]$.
   - Using oracle access to prover $\mathsf{P}'(1^{\kappa}, \cdot)$, compute the value:

$$Z = \sum_{i \in [\varphi] \setminus \{i^*\}} Z_i(\mathbf{B}, \mathbf{C}_{|\mathbf{c}^{i^*,j} = \mathbf{c}}).$$

   If $Z \leqslant k^*$ break the loop and output $\mathbf{a}^{i^*,j}$.
2. Output $\perp$ if no answer has been found.

Notice that the index k^* depends only on the description of P', therefore we can assume that k^* is passed to P^* as auxiliary input. Consider now an execution of the ρ-round protocol between $\mathsf{P}^{*\mathsf{P}'}$ and V on common input x and prover's auxiliary input $z^* := (z, k^*)$, where the prover executes the procedure $\overline{\mathsf{Resp}}_{\tau}$ with oracle access to $\mathsf{P}'(1^{\kappa}, x, z)$ and the verifier samples $(\mathbf{b}, \mathbf{c}) \leftarrow\$ \mathsf{Chall}(1^{\kappa}, x)$. We can upper bound the probability that P^* fails in convincing V as follows:

$$\begin{aligned} \Pr\left[\langle \mathsf{P}^{*\mathsf{P}'}(1^{\kappa}, x, z^*), \mathsf{V}(1^{\kappa}, x)\rangle \neq \mathtt{acc}\right] &\leqslant \Pr\left[\exists j : \overline{\mathsf{Resp}}_{\tau}^{\mathsf{P}'}(1^{\kappa}, x, \mathbf{c}_{\downarrow j}) \neq \mathbf{b}_{\downarrow j}\right] \\ &\leqslant \sum_{j \in [\rho]} \Pr_{\overline{\mathsf{Resp}}_{\tau}, \mathbf{c}}\left[\overline{\mathsf{Resp}}_{\tau}^{\mathsf{P}'}(1^{\kappa}, x, \mathbf{c}_{\downarrow j}) \neq \mathbf{b}_{\downarrow j}\right]. \end{aligned} \tag{2}$$

Next, we analyze the probability that $\overline{\mathsf{Resp}}_{\tau}$ forwards the right answer in a generic round $j \in [\rho]$. The analysis is facilitated by first looking at the algorithm $\overline{\mathsf{Resp}}_{\infty}$ that executes the main loop until it finds a valid answer; later we will show that the probability that the number of executions is less than τ is overwhelming in κ. In case the algorithm $\overline{\mathsf{Resp}}_{\infty}$ ends there are two possible situations:

- The algorithm accepts a wrong answer, let us call this event Fail. In this case, the number of errors made by P' is $k^* + 1$ (including the answer $\mathbf{a}^{i^*,j}$). Observe that, since i^* is uniformly random, at any iteration we have $\Pr[\mathsf{Fail}] = \delta_{\varphi - k^* - 1}\frac{k^*+1}{\varphi}$.
- The algorithm accepts a correct answer, let us call this event $\mathsf{Success}$. In this case, the number of errors made by P' is at most k^*. Observe that, by definition, at any iteration $\Pr[\mathsf{Success}] = \sum_{i \geqslant \varphi - k^*} \delta_i$.

Recalling the definition of the index k^* from Eq. 1, we obtain:

$$\Pr[\mathsf{Fail}] \leqslant \frac{k^* + 1}{\varphi}\delta_{\varphi - k^* - 1} \leqslant \frac{1}{2\rho}\sum_{i \geqslant \varphi - k^*} \delta_i = \frac{1}{2\rho}\Pr[\mathsf{Success}],$$

and thus the probability that $\overline{\mathsf{Resp}}_{\infty}$ fails is less than $\frac{1}{2\rho}$. (We are conditioning on the event that $\overline{\mathsf{Resp}}_{\infty}$ terminates here.)

Since $\Pr[\mathsf{Success}] \geq \delta_{\varphi} = p'$, we have $\Pr[\mathsf{Fail}] + \Pr[\mathsf{Success}] \geqslant p'$. Hence, by the tail inequality of the geometric distribution the probability that $\overline{\mathsf{Resp}}_{\infty}$ makes more than $\tau = {p'}^{-1}\omega(\log\kappa)$ iterations is negligible. Therefore, we can upper bound the probability that $\overline{\mathsf{Resp}}_{\tau}$ fails by $\frac{1}{2\rho} + \mathsf{negl}(\kappa)$. Combining this with Eq. (2) we get that P^* is successful in the ρ-round protocol with probability $1 - \rho(\frac{1}{2\rho} + \mathsf{negl}(\kappa)) \geq 1/3$.

We finally define the f-knowledge extractor of protocol Π' on prover P' to be the same f-knowledge extractor of protocol Π on prover $\mathsf{P}^{*\mathsf{P}'}$.[2] Recall that the knowledge soundness of Π is $\epsilon(\kappa)$; moreover prover P^* succeeds with probability greater than $1/3$ whenever executed with oracle access to a prover P' for Π' that succeeds with noticeable probability. It follows that the knowledge soundness of Π' is $\epsilon^{\omega(\log\kappa)}$.

To complete the proof we show that the completeness error of the collapsed protocol is negligible. Notice that for any $j \in [\rho]$, given an instance $(x, w) \in R$:

$$1 - \mu(\kappa) \geqslant \Pr\left[\langle \mathsf{P}(1^\kappa, x, w), \mathsf{V}(1^\kappa, x)\rangle = \mathtt{acc}\right] = \Pr\left[\wedge_{i\in[\rho]}(a_i = b_i)\right] \geqslant \Pr\left[\wedge_{i\in[j]}(a_i = b_i)\right].$$

Therefore:

$$\Pr\left[\left\langle \mathsf{P}'(1^\kappa, x, w), \mathsf{V}'(x)\right\rangle = \mathtt{acc}\right] \geqslant (1-\mu)^{\rho\cdot\varphi}.$$

Recall that we set $\varphi = \omega(\log\kappa)\rho$ hence the right-hand side of the equation above is overwhelming. □

3.3 On Laconic PAoK

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

Theorem 2. *Let R be an* NP *relation. There exists a PAoK for R if and only if there exists a* laconic *PAoK for R.*

Proof sketch. Consider $\Pi = (\mathsf{Chall}, \mathsf{Resp})$ to be a PAoK for R, with $\ell = \mathsf{poly}(\kappa)$. We write $(\mathbf{c}, \mathbf{b})$ for the output of $\mathsf{Chall}(1^\kappa, x)$ and $\mathbf{a}$ for the output of $\mathsf{Resp}(1^\kappa, x, w, \mathbf{c})$; note that $|\mathbf{a}| = |\mathbf{b}| = \ell$. Define the following laconic protocol $\Pi' = (\mathsf{Chall}', \mathsf{Resp}')$:

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

Clearly, Π' is laconic. Notice that the bit b' is an hard-core predicate [13] for the relation $R_{\mathsf{ip}} = \{(\mathbf{b}, (\mathbf{c}, \mathbf{r})) : (\mathbf{c}, \mathbf{b}) \in \mathsf{Chall}(1^\kappa, x), \mathbf{r} \in \{0,1\}^\ell\}$. Moreover, the inverter of the Goldreich-Levin Theorem [11] works for any surjective function and, for any fixed function $g(b) := (\mathbf{c}, \mathbf{r})$ such that $(\mathbf{b}, (\mathbf{c}, \mathbf{r})) \in R_{\mathsf{ip}}$, it does not require the function $g(\cdot)$ to be efficiently computable in order to invert it. With this in mind, it is not hard to reduce knowledge soundness of Π' to the hardness of the Goldreich-Levin predicate. We skip the technical details.

We proceed to prove the other direction. Let $\Pi' = (\mathsf{Chall}', \mathsf{Resp}')$ be a laconic PAoK for the relation R, and consider the protocol Π^ℓ below which, for a parameter $\ell = \mathsf{poly}(\kappa)$, simply repeats Π' sequentially for $\rho(\kappa)$ times:

1. For $i \in [\rho(\kappa)]$ where $\rho(\kappa) = \omega(\ell^{1/2}/\log^{1/2}\kappa)$, proceed as follows:
   - Execute protocol Π';
   - If the execution aborts then output `rej`, otherwise proceed to the next iteration.
2. If all the ρ executions were accepting, then return `acc`.

It is well known that sequential repetition amplifies the knowledge soundness error of private-coin arguments of knowledge; thus Π^ℓ is a ρ-round PAoK with knowledge soundness error ϵ^ℓ. We can now apply Theorem 1 to obtain a one-round PAoK. □

[2] Strictly speaking, the definition of f-knowledge soundness does not consider provers with oracle access, however, since P' is efficient, we can define a new prover $\hat{\mathsf{P}}$ that executes P^* and emulates each invocation to the oracle using the code of P'.

4 Constructing PAoK

We give two constructions of PAoK. In Section 4.1 we show that we can construct a PAoK from any extractable hash-proof system [24] (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 4.2 we investigate the relationship between PAoK and Extractable Witness Encryption [10, 15]; while Ext-WE for all of NP is implausible [9] we put forward a weaker variant of Ext-WE (weak Ext-WE) which we show to be equivalent to weak PAoK. Since, as we show, weak Ext-WE can be obtained for arbitrary NP relations using *public-coin* differing-inputs obfuscation [18]—see Section 5—we obtain a weak PAoK for all of NP as a corollary.

Finally, in Section 4.3, we use a weak PAoK for NP to construct a PAoK for any random self-reducible relation.

4.1 Construction from Extractable Hash-Proof Systems

The definition below is adapted from [24].

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 $\mathsf{Pub}(1^\kappa, pk, r) = h_{pk}(u)$.

Extraction mode: For all $(pk, sk) \leftarrow \mathsf{SetupExt}(1^\kappa)$ and all (x, y), 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 interactive 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 3. *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-extractable 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 K of the PAoK as follows. Let P^* be a ppt algorithm such that $\langle \mathsf{P}^*(1^\kappa, x, z), \mathsf{V}(1^\kappa, x)\rangle_\Pi = \mathtt{acc}$ with probability $p(\kappa)$, where P^* uses auxiliary input $z \in \{0,1\}^*$. Define $\mathsf{K}(1^\kappa, x, z) := \mathsf{Ext}(1^\kappa, sk, x, a)$ where a is the message sent by P^*. By definition of protocol Π we get that whenever 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 $R'(x, w') = 1$. $\square$

Instantiations. We consider two instantiations of Theorem 3, based on the constructions of Ext-HPS given in [24].

- The first construction is for the Diffie-Hellman relation $R_{\mathsf{DH}} := \{(g^r, g^{\alpha r}) : \ g \in \mathbb{G}, \alpha, r \in \mathbb{Z}_q\}$, for a group $\mathbb{G}$ of prime order q. Note that $\mathsf{SamR}(r) := (g^r, g^{\alpha r})$. The corresponding relation R'_{DH} is $R'_{\mathsf{DH}} := \{(g^r, r) : \ g \in \mathbb{G}, r \in \mathbb{Z}_q\}$, and $f(r) = f_\alpha(r) := g^{\alpha r}$.
- The second construction is based on factoring. Let $R_{\mathsf{QR}} := \{(g^{2^k r}, g^r) : \ g \leftarrow\!\!\$\ \mathbb{QR}_N^+, r \in [(N-1)/4]\}$, where N is a Blum integer. Note that $\mathsf{SamR}(r) := (g^{2^k r}, g^r)$. The corresponding relation R'_{QR} is $R'_{\mathsf{QR}} := \{(g^{2^k r}, r) : \ g \leftarrow\!\!\$\ \mathbb{QR}_N^+, r \in [(N-1)/4]\}$, and $f(r) := g^r$.

4.2 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 [9].

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:[3] (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 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 5 (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 $x \in L$ and $\beta \in \{0,1\}$, we have that $\mathsf{Decrypt}(1^\kappa, w, \mathsf{Encrypt}(1^\kappa, x, \beta)) = \beta$ with probability one, where $(x, w) \in R_L$.

Extractable security: For any ppt adversary $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ and for any noticeable function $\epsilon(\cdot)$, there exists a non-uniform extractor 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 $(x, st) \leftarrow\!\!\$\ \mathsf{A}_0(1^\kappa, z)$, whenever

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

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

Since constructing an Ext-WE scheme for all of NP (w.r.t. arbitrary auxiliary inputs) is implausible [9], a corollary of our analysis below is that full-fledged PAoK for all NP is also implausible. Motivated by this negative result, we define a weaker flavour of Ext-WE (weak Ext-WE) which is similar in spirit to our definition of weak PAoK. As we show, for any relation, we can construct a weak Ext-WE from any weak PAoK for the same relation (and viceversa). Finally we obtain a weak PAoK for NP by noting that any public-coin differing-inputs obfuscator already gives a weak Ext-WE scheme (see Section 5).

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

Definition 6 (Weak Ext-WE)**.** Let R be a relation, and $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ be a WE scheme for R. We say that Π_{WE} is a weak Ext-WE scheme for R if it satisfies correctness as per Definition 5, and if extractable security is replaced by weak extractable security, specified below.

Weak Extractable security: For any ppt adversary A, any ppt algorithm Sam and for any noticeable function $\epsilon(\cdot)$ there exists a non-uniform extractor K and a non-zero polynomial $q(\cdot)$ such that for any auxiliary information $z_A, z_S \in \{0,1\}^{\mathsf{poly}(\kappa)}$ the following holds. Whenever

$$\Pr\left[\mathsf{A}(1^\kappa, r, \mathsf{Encrypt}(1^\kappa, \mathsf{Sam}(1^\kappa, z_S; r), \beta), z_A) = \beta : \ \beta \leftarrow_\$ \{0,1\}\right] \geqslant \frac{1}{2} + \epsilon(\kappa)$$

then $\Pr[(x,w) \in R_L : \ x := \mathsf{Sam}(1^\kappa, z_S; r), w \leftarrow_\$ \mathsf{K}(1^\kappa, x, r, z_A, z_S)] \geqslant q(\epsilon(\kappa))$.

Theorem 4. *Let R be an NP-relation. There exists a (weak) PAoK for R if and only if there exists a (weak) Ext-WE scheme for R.*

Proof. 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 $a \leftarrow \mathsf{Resp}(1^\kappa, x, w, c)$.

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

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

for $(st, x) \leftarrow_\$ \mathsf{A}_0(1^\kappa, z)$ (where $z \in \{0,1\}^*$ is the auxiliary input). We use A to construct a prover P^* attacking knowledge soundness of Π. Prover P^* first runs $(st, x) \leftarrow_\$ \mathsf{A}_0(1^\kappa, z)$, 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 $\mathsf{A}_1(1^\kappa, st, x, \gamma, z)$ 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 P^* has the right distribution (in particular, the second component is a random bit). Since $\beta' = \beta \oplus b$ we get that P^* outputs $a = b$ with probability at least $1/2 + \epsilon(\kappa)$ and thus P^* convinces V with probability $p(\kappa) \geq 1/2 + \epsilon(\kappa)$. We are now in a position to run the extractor K of Π, and hence we obtain a valid witness $w \leftarrow_\$ \mathsf{K}(1^\kappa, x, z)$ with probability $q(\epsilon(\kappa))$. The statement follows. A similar argument shows that Π_{WE} is a weak Ext-WE whenever Π is a weak PAoK.

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 P^* be a malicious prover for the PAoK. Assume that there exists a polynomial $p(\cdot)$ such that

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

where $z \in \{0,1\}^*$ is the auxiliary input. We use P^* to construct an adversary $\mathsf{A} := (\mathsf{A}_0, \mathsf{A}_1)$ attacking extractable security of Π_{WE}. Adversary $\mathsf{A}_0(1^\kappa, z)$ outputs x, and then 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 A proceeds as follows:

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

For the analysis, note that the challenge simulated by 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 A_1 guesses β with at least the same probability. We are now in a position to run the extractor K of Π_{WE}, and hence we obtain a valid witness $w \leftarrow\!\!\$\ \mathsf{K}(1^\kappa, x, z)$ with probability $q(p(\kappa) - \epsilon(\kappa))$. The statement follows. A similar argument shows that Π is a weak PAoK whenever Π_{WE} is a weak Ext-WE. $\square$

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., [1, 23, 20]).

4.3.1 Random Self Reducibility

Informally a function is random self-reducible if, given an algorithm that computes the function on random instances, 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 7 (Self-Reducible Relation)**.** An NP-relation R for a language L is random self-reducible if there exists a pair of ppt algorithms $(\mathsf{W}_0, \mathsf{W}_1)$ 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 $x' := \mathsf{W}_0(x; \omega)$ where $\omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{poly}(|x|)}$ and set $w' := \mathcal{O}_R(x')$;
  - Let $w := \mathsf{W}_1(x, w'; \omega)$;

  Then $(x, w) \in R$.
- The value x' is uniformly distributed over L.

We call the pair of algorithms $\mathsf{W} = (\mathsf{W}_0, \mathsf{W}_1)$ an average-to-worst-case (AW) reduction.

Notice that the reduction W has oracle access to a "powerful" oracle that produces a witness for a randomized instance, and uses this witness to compute a witness for the original instance. As a toy 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 $z \in \mathbb{Z}_q$, compute $x' = x \cdot g^z$ and ask to the oracle a witness for x'; (ii) Given w' such that $g^{w'} = x'$ compute $w := w' - z$.

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

Definition 8 (Witness Re-constructibility)**.** A random self-reducible relation R with AW reduction $\mathsf{W} = (\mathsf{W}_0, \mathsf{W}_1)$ is *witness reconstructible* for W if there exists a ppt algorithm $\overline{\mathsf{W}}$ such that for any $\omega \in \{0,1\}^{\mathsf{poly}(|x|)}$ and for any $(x, w) \in R$ the following holds: Let x' be the oracle call made by $\mathsf{W}(x; \omega)$, and define $w' := \overline{\mathsf{W}}(x, w; \omega)$; then $(x', w') \in R$.

4.3.2 The protocol

We show how to use a *weak* PAoK 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 AW reduction $\mathsf{W} = (\mathsf{W}_0, \mathsf{W}_1)$. Let $\Pi' := (\mathsf{Chall}', \mathsf{Resp}')$ be a weak PAoK for R. Consider the following protocol $\Pi = (\mathsf{Chall}, \mathsf{Resp})$:

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

Theorem 5. *Let R be a random self-reducible* NP*-relation which is witness reconstructible and has AW reduction $\mathsf{W} = (\mathsf{W}_0, \mathsf{W}_1)$. Let Π' be a weak PAoK for R, with knowledge error ϵ. Then protocol Π described above is a PAoK for R with knowledge error ϵ.*

Proof. For any ppt prover P^* we need to define a knowledge extractor K such that for any instance x for which $p(\kappa) := \Pr[\langle \mathsf{P}^*(1^\kappa, x, z), \mathsf{V}(1^\kappa, x)\rangle = \mathtt{acc}] > \epsilon(\kappa)$ the extractor K produces a witness w for x with probability $q(p(\kappa) - \epsilon(\kappa))$ for an inverse-polynomial function $q(\cdot)$. Let K' be the weak-knowledge extractor of Π', for prover P^* and sampler W_0, and denote by $q'(\cdot)$ the corresponding inverse-polynomial function. Notice that for any $z_P \in \{0,1\}^*$ the extractor $\mathsf{K}'(1^\kappa, x, \omega, z_P)$ produces a witness for $x' = \mathsf{W}_0(x;\omega)$ with probability $q'(p(\kappa) - \epsilon(\kappa))$ where the probability is over the choice of ω and over the random coins of of K'. Consider the knowledge extractor K that works as follow:

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

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

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

□

5 Weak PAoK for NP

We recall the notion of public-coin differing-inputs obfuscation. This notion was proposed by Ishai *et al.* in [18] and it avoids the negative result of [9] by imposing the restriction that two almost functionally-equivalent circuits can be safely obfuscated only if it is hard to find an input where they differ even given the randomness that produced the two circuits.

We first need to define public-coin differing-inputs samplers.

Definition 9 (Public-Coin Differing-Inputs Sampler for Circuits)**.** An efficient non-uniform sampling algorithm $\mathsf{Sam} = \{\mathsf{Sam}_\kappa\}_{\kappa \in \mathbb{N}}$ is called a public-coin differing-inputs sampler for the parametrized collection of circuits $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ if the output of Sam_κ is distributed over $\mathcal{C}_\kappa \times \mathcal{C}_\kappa$ and for every efficient non-uniform algorithm A there exists a negligible function ϵ such that for all $\kappa \in \mathbb{N}$:

$$\Pr_r[C_0(x) \neq C_1(x) :\ (C_0, C_1) \leftarrow\!\!\$\ \mathsf{Sam}_\kappa(r), x \leftarrow\!\!\$\ \mathsf{A}_\kappa(r)] \leqslant \epsilon(n).$$

While reading the definition below, keep in mind that the sampler cannot "keep any secret" from the adversary.

Definition 10 (Public-Coin Differing-Inputs Obfuscator for Circuits)**.** A uniform ppt algorithm diO is a public-coin differing-inputs obfuscator for the parametrized collection of circuits $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ if the following requirements are met.

- **Correctness:** $\forall\kappa, \forall C \in \mathcal{C}_\kappa, \forall x$ we have that $\Pr[C'(x) = C(x) :\ C' \leftarrow\!\!\$\ \mathsf{diO}(1^\kappa, C)] = 1$.
- **Security:** For every public-coin differing-inputs sampler $\mathsf{Sam} = \{\mathsf{Sam}_\kappa\}_{\kappa\in\mathbb{N}}$ for the collection $\mathcal{C}$, every efficient non-uniform (distinguishing) algorithm $\mathsf{D} = \{\mathsf{D}_\kappa\}_{\kappa\in\mathbb{N}}$, there exists a negligible function ϵ such that for all $\kappa \in \mathbb{N}$:

$$\Big| \Pr[\mathsf{D}_\kappa(r, C') = 1 :\ (C_0, C_1) \leftarrow\!\!\$\ \mathsf{Sam}_\kappa(r), C' \leftarrow\!\!\$\ \mathsf{diO}(1^\kappa, C_0)] - \Pr[\mathsf{D}_\kappa(r, C') = 1 :\ (C_0, C_1) \leftarrow\!\!\$\ \mathsf{Sam}_\kappa(r), C' \leftarrow\!\!\$\ \mathsf{diO}(1^\kappa, C_1)] \Big| \leqslant \epsilon(\kappa),$$

  where the probability is over the choice of r and the coins of diO.

Consider the following construction of a WE scheme $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ for any relation R.[4]

- Upon input $1^\kappa, x$ and message $\beta \in \{0,1\}$, define $\mathsf{Encrypt}(1^\kappa, x, \beta) := \mathsf{diO}(C_{x,\beta}) := \gamma$ where $C_{x,\beta}$ is the circuit that hard-wires x and β and, upon input a value w, it returns β iff $(x, w) \in R$ (and otherwise $\perp$).
- Upon input $1^\kappa, x, w, \gamma$, define $\mathsf{Decrypt}(1^\kappa, x, w, \gamma) = \gamma(w)$.

Theorem 6. *If* diO *is a public-coin differing input obfuscator then the scheme* Π_{WE} *described above is a weak Ext-WE scheme for the relation* R*.*

Proof. Suppose that Π_{WE} is not a weak Ext-WE. This means there exists a ppt sampler Sam', a ppt adversary A, and a noticeable function $\epsilon(\cdot)$ such that for infinitely many κ there exist auxiliary informations $z_A, z_S \in \{0,1\}^{\mathsf{poly}(\kappa)}$ for which

$$\Pr\left[\mathsf{A}(1^\kappa, r, \mathsf{Encrypt}(1^\kappa, \mathsf{Sam}'(1^\kappa, z_S; r), \beta), z_A) = \beta :\ \beta \leftarrow\!\!\$\ \{0,1\}\right] \geqslant \frac{1}{2} + \epsilon(\kappa)$$

but where for any non-uniform extractor K :

$$\Pr\left[(x, w) \in R_L :\ \begin{array}{c} x := \mathsf{Sam}'(1^\kappa, z_S; r), \\ w \leftarrow\!\!\$\ \mathsf{K}(1^\kappa, x, r, z_A, z_S) \end{array}\right] \leqslant \mathsf{negl}(\kappa). \tag{3}$$

It is not hard to see that from the equation above, we can derive a public-coin differing-input sampler for the circuits $(C_{x,0}, C_{x,1})$. Specifically, let $\mathsf{Sam} = \{\mathsf{Sam}_\kappa\}_{\kappa\in\mathbb{N}}$ be the following non-uniform machine where Sam_κ on input randomness r: (i) Sets $x := \mathsf{Sam}'(1^\kappa, z_s; r)$; (ii) Outputs $(C_{x,0}, C_{x,1})$. Notice that

[4] Recall that, as shown in Theorem 4, given a weak Ext-WE for a relation we can construct a weak PAoK for the same relation.

$C_{x,0}, C_{x,1}$ differ only in the inputs w such that $(x, w) \in R$, therefore, to keep the syntax coherent,[5] for any non-uniform machine A':

$$\Pr_r \left[C_{x,0}(w) \neq C_{x,1}(w) : \begin{array}{c} (C_{x,0}, C_{x,1}) := \mathsf{Sam}_\kappa(r), \\ w \leftarrow\!\!\$\ \mathsf{A}'_\kappa(r) \end{array} \right] \leqslant \mathsf{negl}(\kappa).$$

However an encryption of β is just $\mathsf{diO}(C_{x,\beta})$, therefore we obtain that $\mathsf{D}_\kappa(r, C') := \mathsf{A}(1^\kappa, r, C', z_A)$ is a distinghuisher for diO. This contradicts the assumption that diO is a public-coin differing-inputs obfuscator.
□

References

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

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

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

[4] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *TCC*, pages 52–73, 2014.

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

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

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

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

[9] 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.

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

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

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

[5] Notice that since Eq. (3) holds for any non-uniform K with input $1^\kappa, x, r, z_A, z_S$, then it holds also for any A' with input r.

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

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

[15] 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.

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

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

[18] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In *TCC*, pages 668–697, 2015.

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

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

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

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

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

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

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

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

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

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

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

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.* [3]—for secure message transmission. Informally, in a leakage-tolerant message transmission protocol leakage queries from an adversary A are viewed as a form of partial corruptions, where 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 [5]. 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 ideal functionality $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$. We denote with $\mathbf{IDEAL}_{\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}, \mathsf{S}, \mathsf{Z}}(\Phi, \kappa)$ 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

$\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$ 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 11 (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 $\mathbf{F}_{\mathsf{SMT}}^{+\mathsf{lk}}$, 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

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

Necessity of long keys. Nielsen *et al.* [19] have shown that any leakage-tolerant PKE protocol (as per Definition 11) 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 $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$:

$$R_{\mathsf{PKE}} := \{((pk, \gamma, m), (sk, r_G)) : \ (pk, sk) = \mathsf{KGen}(1^\kappa; r_G) \wedge \mathsf{Dec}(sk, \gamma) = m\} \tag{4}$$

We show the following theorem.

Theorem 7. *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 7. When the knowledge soundness of the PAoK is negligible and the PKE protocol is (ℓ, 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 [19].

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 $\phi^{x,c}_{\mathsf{Z}}$ 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 $\epsilon = 2^{-\ell} + \mathsf{negl}(\kappa)$. 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 [19, Theorem 1] shows that $m \in M_{pk,\gamma}$ except with probability $\epsilon + \epsilon'$. Combined with the above, this implies that $|M_{pk,\gamma}| \geq (1 - 2^{-\ell} - \mathsf{negl}(\kappa) - 2\epsilon')|\mathcal{M}|$. 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. $\square$