

PERK: Compact Signature Scheme Based on a New Variant of the Permuted Kernel Problem

Slim Bettaieb¹, Loïc Bidoux¹ , Victor Dyseryn², Andre Esser^{1*} , Philippe Gaborit², Mukul Kulkarni¹, and Marco Palumbi¹

¹ Technology Innovation Institute, UAE
{slim.bettaieb, loic.bidoux, andre.esser,
mukul.kulkarni, marco.palumbi}@tii.ae
² XLIM, University of Limoges, France
{victor.dyseryn_fostier, gaborit}@unilim.fr

Abstract. In this work we introduce PERK a compact digital signature scheme based on the hardness of a new variant of the Permuted Kernel Problem (PKP). PERK achieves the smallest signature sizes for any PKP-based scheme for NIST category I security with 6 kB, while obtaining competitive signing and verification timings. PERK also compares well with the general state-of-the-art. To substantiate those claims we provide an optimized constant-time AVX2 implementation, a detailed performance analysis and different size-performance trade-offs.

Technically our scheme is based on a Zero-Knowledge Proof of Knowledge following the MPC-in-the-Head paradigm and employing the Fiat-Shamir transform. We provide comprehensive security proofs, ensuring EUF-CMA security for PERK in the random oracle model. The efficiency of PERK greatly stems from our particular choice of PKP variant which allows for an application of the challenge-space amplification technique due to Bidoux-Gaborit (C2SI 2023).

Our second main contribution is an in-depth study of the hardness of the introduced problem variant. First, we establish a link between the hardness of our problem variant and the hardness of standard PKP. Then, we initiate an in-depth study of the concrete complexity to solve our variant. We present a novel algorithm which outperforms previous approaches for certain parameter regimes. However, the proximity of our problem variant to the standard variant can be controlled via a specific parameter. This enables us to effectively safeguard against our new attack and potential future extensions by a choice of parameters that ensures only a slight variation from standard PKP.

Keywords: Post-Quantum Cryptography · Cryptanalysis · New Hardness Assumption · Digital Signatures · PKP · PoK · MPC

* supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID MA 2536/12

1 Introduction

The rising threat posed by quantum algorithms to classical cryptographic schemes following the seminal work of Shor [52], has led to a significant interest in post-quantum cryptography research. This shift towards study of quantum secure hardness assumptions [3, 19, 28, 49] and their use in cryptographic system design [14, 43, 46] has captured the attention of the cryptographic community in recent years. This trend has been further supported by NISTs standardization efforts for post-quantum secure schemes initiated back in 2017 [41]. This initial call targeted Key Encapsulation Mechanisms (KEMs) as well as digital signature schemes. While for both classes final standards have been selected by now, the KEM process entered an additional 4th round for further evaluation of existing candidates. The goal of that additional round is to enrich the set of available standards. For digital signature schemes this was not possible due to the lack of suitable candidates. Overall, this led to a current NIST signature portfolio that either lacks diversity in its hardness assumptions by being entirely relying on structured lattice-based assumptions [40, 48] or suffers certain performance compromises [32].

These circumstances led to a reopening of the call for proposal for digital signature schemes with main goal of adding general purpose signature schemes that are not based on computational hardness assumptions related to structured lattices [42]. In this work we present PERK a new post-quantum secure digital signature scheme based on a variant of the Permuted Kernel Problem. PERK is constructed from a zero-knowledge proof of knowledge (ZKPoK) using the Fiat-Shamir transformation [26] following the MPCitH (MPC in the Head) paradigm [33, 34]. It achieves competitive performances in terms of signature size and signing as well as verification time, well addressing the current need for general purpose post-quantum secure digital signature schemes.

Digital signatures from Proofs of Knowledge. A ZKPoK is an interactive protocol allowing a prover to convince a verifier that it knows a certain secret without revealing any information on the secret. The Fiat-Shamir transformation allows to convert any such ZKPoK into a non-interactive digital signature scheme which is secure in the random oracle model. ZKPoKs are generally not perfectly sound, i.e., there is a non-zero probability that a malicious prover may succeed in convincing the verifier without actually knowing the secret. Therefore, in order to obtain a secure digital signature scheme the underlying ZKPoK is repeated several times which hampers the performance of such constructions. Naturally, the community has focused on techniques to improve soundness guarantees provided by the underlying ZKPoK protocol. In 2018, Katz, Kolesnikov and Wang [37] proposed a protocol based on the MPC-in-the-Head paradigm of [33, 34] in the preprocessing model. This technique has notably been leveraged by the NIST candidate Picnic [17]. Two years later, the “PoK with Helper” paradigm was introduced by Beullens [12]. In the PoK-with-Helper paradigm, the helper plays the role of a trusted third party which facilitates the design of PoK and is finally replaced by the well-known cut-and-choose technique when instantiating

the protocol. The PoK-with-Helper paradigm leads to shorter signature sizes at the cost of a performance overhead. Bidoux and Gaborit [15] proposed an alternate framework to replace the Helper thus reducing the associated performance overhead. They suggested to use variants of the underlying hard computational problems which are more suitable for the design of efficient MPC protocols. Recently, many constructions leveraging the MPC-in-the-Head paradigm have been proposed in the literature [1, 2, 7, 15, 16, 18, 21, 22, 24, 25, 31, 53]. Most of these constructions are 5-round protocols hence must take into account the attack of [36] when selecting parameters.

The Permuted Kernel Problem. The ZKPoK from which our scheme is constructed proves knowledge to the solution of (a slight variant of) the Permuted Kernel Problem (PKP). The PKP was introduced by Shamir in his pioneering work in 1990 [51]. Since its introduction, many variants and natural extensions of the problem and their hardness have been studied [39]. In our work we retain one of the most general formulations of the problem. Informally, this problem asks, given a matrix \mathbf{H} and t vectors \mathbf{x}_i , to find a permutation π that sends $\pi[\mathbf{x}_i]$ into the Kernel of \mathbf{H} for all i , i.e., a permutation such that $\mathbf{H}\pi[\mathbf{x}_i] = \mathbf{0}$.

The PKP for $t = 1$ has been proven NP hard in [29]. Recently, the more general formulation with $t \geq 1$ has also been shown to be NP hard through a reduction to the subcode equivalence problem [10, 50]. Despite many cryptanalytic efforts over the years [8, 30, 35, 38, 39, 45, 50] the problem remains hard against classical as well as quantum attackers. The well-studied hardness, compact description and simplicity of involved objects and corresponding computations has made the PKP an attractive candidate for post-quantum secure schemes in recent years [12, 13, 15].

Our construction PERK relies on a non-trivial application of the challenge-space amplification technique from [15]. In order to do so, we introduce a slightly relaxed variant of the PKP problem denoted r -IPKP. We provide security arguments that relate its hardness to the hardness of standard PKP and additionally provide a detailed study of its concrete complexity. We also present a new algorithm for solving the problem variant that outperforms other approaches for certain parameter regimes. To the best of our knowledge, PERK obtains the smallest signatures among all PKP-based signature schemes. We provide security proofs for our scheme as well as a constant-time implementation. Our scheme outperforms SPHINCS⁺ (the only non lattice based signature selected by the NIST) in both size and signing time. For NIST category I security, our construction features a signature size ~ 2 kB shorter and a signing algorithm ~ 6 times faster than SPHINCS⁺ (however unsurprisingly verification is faster for SPHINCS⁺).

Our contributions. The main contributions of our work can be detailed as follows:

- We introduce a new slightly relaxed version of PKP, which we call r -IPKP (see Definition 2.9). Furthermore, we present compelling arguments that establish a connection between the complexity of r -IPKP and that of standard

PKP. Specifically, we demonstrate that solving r -IPKP can be achieved by applying any PKP algorithm or by resolving one among several provided PKP instances with $t = 1$.

- We initiate a detailed study of the hardness of r -IPKP. To this end we first conduct a thorough survey of existing attacks against PKP applicable to r -IPKP. Furthermore, we present and analyze a new algorithm for solving r -IPKP that exploits the established link to the multi-instance version of PKP. This algorithm outperforms standard PKP approaches when either many \mathbf{x}_i are given, i.e. for high values of t , or if many solutions exist.
- We design a new ZKPoK for the r -IPKP problem. Using our newly introduced problem allows to leverage the challenge space amplification technique from [15] and reduce the impact of the [36] attack thus leading to an efficient construction. Finding and defining a natural relaxation of a given problem that can be beneficial for a MPC based PoK while being easy to analyze and retaining the security properties of the initial problem is a delicate task. Analogous problem relaxations have been studied in other contexts (such as using the Rank Support Learning (RSL) problem [27] instead of the Rank Syndrome Decoding (RSD) problem) however this is the first time that such an idea is applied to the PKP setting.
- We then build a new post-quantum digital signature scheme (PERK) from this ZKPoK. PERK greatly improves on previous PKP-based schemes with a signature size of 6 kB for NIST category I security (in comparison to 8.9 kB achieved in [15]). To the best of our knowledge PERK therefore achieves the smallest signatures among PKP-based schemes.
- We also provide an optimized constant-time AVX2 implementation of our scheme. This implementation allows signing in 36 million cycles (12 ms) and verification in 25 million cycles (8 ms) (for NIST category I security, with 6kB signatures). Furthermore, the scheme allows for interesting size/performance trade-offs, as for example signing and verification can be sped up by a factor of 5 with only a slight increase in signature size.

It should be pointed out that our parameter selection process is very conservative. We ignore any polynomial factors in the complexity estimation and match the NIST security thresholds of 143, 207 and 272 bit rather than the commonly used thresholds of 128, 192 and 256 bit. Furthermore, we ignore any cost that stems from memory access, even though all considered algorithms have a high memory demand, and make optimistic assumptions on possible speedups from multiple solutions. Still, our scheme is very competitive in terms of both, size as well as speed, when compared to the state of the art (for details see Tables 5 and 6). Additionally, the new problem variant r -IPKP might be of independent interest for designing post-quantum cryptographic schemes as well as for cryptanalytic efforts.

Outline. In Section 2 we cover basic notations and definitions, recall ZKPoK constructions as well as the MPCitH paradigm and introduce the different notions of the PKP. In the subsequent Section 3 we then present our new ZKPoK

protocol based on the hardness of the r-IPKP, convert it into a non-interactive signature scheme via the Fiat-Shamir transform and provide comprehensive security proofs. Section 4 covers a detailed analysis of the complexity of r-IPKP, where Section 4.1 covers previous approaches and Section 4.2 introduces our new algorithm. Parameters of our scheme are presented in Section 5 which also provides a comparison to the state-of-the-art. Eventually, in Section 6 performance benchmarks of our optimized AVX2 constant-time implementation are given.

2 Preliminaries

2.1 Notations and Conventions

For integers a, b we denote $[a, b]$ the set of integers i such that $a \leq i \leq b$. We write $[n]$ as a shorthand for $[1, n]$. We denote \mathcal{S}_n the group of permutations of the set $[n]$. Let \mathbb{F}_q denote the finite field of q elements where q is the power of a prime. Vectors are denoted by bold lower-case letters and matrices by bold capital letters (e.g., $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{F}_q^n$ and $\mathbf{M} = (m_{ij})_{\substack{1 \leq i \leq k \\ 1 \leq j \leq n}} \in \mathbb{F}_q^{k \times n}$).

If S is a finite set, we denote by $x \xleftarrow{\$} S$ that x is chosen uniformly at random from S . Similarly, we write $x \xleftarrow{\$, \theta} S$, if x is sampled pseudo-randomly from the set S , based on the seed θ .

We use x to denote input and denote its length by $|x|$. We use λ to denote the security parameter. We call a function $f : \mathbb{N} \rightarrow \mathbb{R}^+$ *negligible*, if for all $c \in \mathbb{N}$ there exists a $N_0 \in \mathbb{N}$ such that $f(n) < 1/n^c$ for all $n > N_0$. We write $\text{negl}(\lambda)$ to denote an arbitrary negligible function. We use $\text{poly}(\lambda)$ for function which is *polynomially bounded* in λ , that is there exists $c, \lambda_0 \in \mathbb{N}$ such that $\text{poly}(\lambda) \leq \lambda^c$ for all $\lambda \geq \lambda_0$. We also abbreviate *probabilistic polynomial-time* as PPT.

Let X and Y be two discrete random variables defined over a finite support D . The *statistical distance* between the two distributions is defined as

$$\Delta(X, Y) := \frac{1}{2} \sum_{d \in D} |\Pr[X = d] - \Pr[Y = d]|.$$

We say two ensembles of random variables $\{X_\lambda\}_{\lambda \in \mathbb{N}}, \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically close* if there exists a negligible function $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\Delta(X_\lambda, Y_\lambda) \leq \text{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$. We say two ensembles of random variables $\{X_x\}_{x \in \{0,1\}^*}, \{Y_x\}_{x \in \{0,1\}^*}$ are *statistically close* if there exists a negligible function $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$ such that $\Delta(X_x, Y_x) \leq \text{negl}(|x|)$ for all $x \in \{0,1\}^*$.

We present the definitions of standard cryptographic primitives such as pseudorandom generators (PRG), collision-resistant hash functions (CRHF), commitment schemes, and Merkle trees in Section A.1 of the supplementary material.

2.2 Zero-Knowledge Proofs of Knowledge

We now present the definitions related to zero-knowledge proofs of knowledge. We closely follow the presentation of these concepts given in [4–6].

Let $R \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be a binary relation. If $(x; w) \in R$, we say x is a *statement* and w is a *witness* for x . The set of valid witnesses for x is denoted by $R(x) = \{w \mid (x; w) \in R\}$. A statement that admits a witness is called a *true* or *valid* statement. The set of true statements is denoted by $L_R := \{x : \exists w \text{ such that } (x; w) \in R\}$. A binary relation is said to be an **NP** relation if the validity of a witness w can be verified in time polynomial in the size $|x|$ of the statement x . From now on we assume all relations to be NP relations.

Informally, an interactive proof system is a protocol involving exchange of messages between a prover and a verifier, through which the prover aims to convince the verifier of the validity of some statement. A proof system is said to be *complete* if the verifier accepts the prover's claim for a valid (or true) statement. And a proof system is said to be *sound* if a cheating prover cannot convince a verifier of the validity of a false statement. We give the formal definitions related to interactive proof systems in Section A.2 of the supplementary material. In the following we only provide the necessary definitions of knowledge soundness, special soundness, and special honest-verifier zero knowledge properties of the interactive proof systems.

Definition 2.1 (Knowledge Soundness (cf. [5])). *An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for relation R is knowledge sound with knowledge error $\varepsilon_{\text{KS}} : \{0, 1\}^* \rightarrow [0, 1]$ if there exists a positive polynomial q and an algorithm Ext , called knowledge extractor, with the following properties: The extractor Ext , given input x and rewindable oracle access to a (potentially dishonest) prover P^* , runs in an expected number of steps that is polynomial in $|x|$ and outputs a witness $w \in R(x)$ with probability*

$$\Pr\left[\left(x; \text{Ext}^{\mathsf{P}^*}(x)\right) \in R\right] \geq \frac{\epsilon(x, \mathsf{P}^*) - \varepsilon_{\text{KS}}(x)}{q(|x|)},$$

where $\epsilon(x, \mathsf{P}^*) := \Pr[(\mathsf{P}^*, \mathsf{V})(x) = \text{accept}]$.

If $\epsilon(x, \mathsf{P}^*) = \Pr[(\mathsf{P}^*, \mathsf{V})(x) = \text{accept}] > \varepsilon_{\text{KS}}(x)$, then the success probability of the knowledge extractor Ext in Definition 2.1 is positive. Therefore, $\epsilon(x, \mathsf{P}^*) > \varepsilon_{\text{KS}}(x)$ implies that x admits a witness, i.e., $x \in L_R$. Hence, knowledge soundness implies soundness.

Definition 2.2 (Proof of Knowledge (cf. [5])). *An interactive proof that is both complete with completeness error $\rho(\cdot)$ and knowledge sound with knowledge error $\varepsilon_{\text{KS}}(\cdot)$ is a Proof of Knowledge (PoK) if there exists a polynomial q such that $1 - \rho(x) \geq \varepsilon_{\text{KS}}(x) + 1/q(|x|)$ for all x .*

It is desirable to have simple verifiers which can send uniform random challenges to the prover, and efficiently verify the transcript.

Definition 2.3 (Public-Coin (cf. [6])). *An interactive proof $\Pi = (\mathsf{P}, \mathsf{V})$ is public-coin if all of V 's random choices are made public, i.e. are part of the transcript. The message $\text{ch}_i \xleftarrow{\$} \mathcal{CH}_i$ of V in the $2i$ -th round is called the i -th challenge, and \mathcal{CH}_i is the challenge set.*

Public-coin protocols can be turned into non-interactive protocols by using the Fiat-Shamir transformation [26]. In this work, we consider only public-coin protocols.

Next, we discuss the notion of *special-soundness*. Special-soundness property is easier to check than knowledge soundness and for many protocols knowledge soundness follows from special-soundness. Note that this requires special-sound protocols to be public-coin.

Definition 2.4 (*k-out-of-N Special Soundness (cf. [5])*). *Let $k, N \in \mathbb{N}$. A 3-round public-coin protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for relation R , with challenge set of cardinality $N \geq k$, is *k-out-of-N special sound* if there exists a polynomial time algorithm that, on input a statement x and k accepting transcripts $(\text{cmt}, \text{ch}_1, \text{rsp}_1), \dots, (\text{cmt}, \text{ch}_k, \text{rsp}_k)$ with common first message cmt and pairwise distinct challenges $\text{ch}_1, \dots, \text{ch}_k$, outputs a witness $w \in R(x)$. We also say Π is *k-special-sound* and, if $k = 2$, it is simply called *special-sound*.*

The definition of special-soundness given in Definition 2.4 can be generalized to protocols where the challenges ch_i are drawn from challenge sets with different cardinalities. We present these definitions in Section A.2 of the supplementary material, and here present only the following theorem proved in [4] which states that special soundness implies knowledge soundness.

Theorem 2.1 (*(k_1, \dots, k_μ) Special Soundness implies Knowledge Soundness [4, Theorem 1]*). *Let $\mu, k_1, \dots, k_\mu \in \mathbb{N}$ be such that $K = \prod_{i=1}^\mu k_i$ can be upper bounded by a polynomial. Let (P, V) be a (k_1, \dots, k_μ) special sound $(2\mu + 1)$ -round interactive protocol for relation R , where V samples each challenge uniformly at random from a set of cardinality N_i for $1 \leq i \leq \mu$. Then (P, V) is knowledge sound with knowledge error*

$$\varepsilon_{\text{KS}} = \frac{\prod_{i=1}^\mu N_i - \prod_{i=1}^\mu (N_i - k_i + 1)}{\prod_{i=1}^\mu N_i} \leq \sum_{i=1}^\mu \frac{k_i - 1}{N_i} \quad (1)$$

We write $\Pi^\tau := (\mathsf{P}^\tau, \mathsf{V}^\tau)$ for the τ -fold parallel repetition of Π , which runs τ instances of Π in parallel and the verifier V^τ accepts if *all* the parallel instances are accepted.

The following theorem proved in [5] states that the knowledge soundness is retained (and knowledge error is reduced) via parallel repetition.

Theorem 2.2 (*Parallel Repetition for Multi-Round Protocols [5, Theorem 4]*). *Let (P, V) be a (k_1, \dots, k_μ) -out-of- (N_1, \dots, N_μ) special sound protocol. Let $(\mathsf{P}^\tau, \mathsf{V}^\tau)$ be the τ -fold repetition of protocol (P, V) . Then $(\mathsf{P}^\tau, \mathsf{V}^\tau)$ is knowledge sound with knowledge error $\varepsilon_{\text{KS}}^\tau$, where*

$$\varepsilon_{\text{KS}} = 1 - \prod_{i=1}^\mu \frac{(N_i - k_i + 1)}{N_i} \quad (2)$$

is the knowledge error of (P, V) .

Definition 2.5 (Special Honest-Verifier Zero Knowledge (SHVZK) (adapted from [4])). *An interactive proof $\Pi = (P, V)$ is called { perfectly, statistically, computationally } honest-verifier zero knowledge (HVZK) if there exists a polynomial time simulator that on input $x \in L_R$ outputs an accepting transcript which is distributed { perfectly, statistically, computationally } close to the transcripts generated by honest executions of Π . If the simulator proceeds by first sampling the verifier’s messages uniformly at random, then Π is called special honest-verifier zero knowledge (SHVZK).*

2.3 MPC-in-the-Head and PoK

Our construction relies on the *MPC-in-the-Head* (MPCitH) paradigm introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai in [33, 34]. This paradigm builds a zero-knowledge proof based on a secure multiparty computation (MPC) protocol. Informally, the MPC protocol is used to compute the verification of an NP relation, where the privacy guarantee of the protocol is used to achieve the zero-knowledge property.

The main steps of the proof of knowledge resulting from the MPCitH technique are the following:

1. The prover splits its witness into N parties by secret sharing the witness;
2. The prover then simulates locally (“in her head”) all the parties of the MPC protocol which evaluates a boolean function that is expected to be 1 whenever the witness is correct (this is supposed to correspond to the verification of desired NP relation);
3. The prover commits to the views of all the parties in the MPC protocol;
4. The verifier chooses a random subset of $N' < N$ parties and asks to reveal their corresponding views;
5. The verifier finally checks that the views of the revealed parties are consistent with each other and with an honest execution of the MPC protocol that yields output 1.

This transformation achieves the zero-knowledge property as long as the views of any N' parties do not leak any information about the secret witness.

Since our proof of knowledge is an instantiation of the MPCitH technique for the specific case of r-IPKP, it benefits from an extensive literature of optimizations generic to any MPCitH construction, such as:

- The preprocessing extension, introduced in [37], allows the MPC protocol – used in the MPCitH technique – to rely on a preprocessing phase (under certain conditions) thus drastically reducing the proof size;
- the challenge space amplification technique, introduced in [15], that is itself an optimization of the PoK with Helper paradigm introduced in [12];
- Merkle trees to reveal a partial number of random seeds, as explained in Section A.1.1 of the supplementary material.

Interactive proof of knowledge can be converted into digital signature schemes using Fiat-Shamir transformation [26] in the random oracle model. We next give the definitions related to signature schemes and present the details related to random oracle model, and Fiat-Shamir transform in Section A.3 of the supplementary material.

Definition 2.6 (Signature Scheme). *A signature scheme consists of three probabilistic polynomial time algorithms $(\text{KeyGen}, \text{Sign}, \text{Vf})$ which work as follows:*

- $\text{KeyGen}(1^\lambda)$: *The key generation algorithm takes a security parameter as input and outputs a pair of keys (pk, sk) . The key sk is the private (secret) signing key and pk is the public key used for verification.*
- $\text{Sign}_{\text{sk}}(m)$: *The signing algorithm takes as input a secret signing key sk and a message m from some message space (that may depend on pk). It outputs a signature $\sigma \leftarrow \text{Sign}_{\text{sk}}(m)$.*
- $\text{Vf}_{\text{pk}}(m, \sigma)$: *The deterministic verification algorithm takes as input a public key pk , a message m , and a signature σ . It outputs a bit $b := \text{Vf}_{\text{pk}}(m, \sigma)$, with $b = 1$ meaning the signature-message pair is valid and $b = 0$ meaning it is invalid.*

Definition 2.7 (EUF-CMA Security). *A signature scheme $(\text{KeyGen}, \text{Sign}, \text{Vf})$ is EUF-CMA secure if, for all PPT adversaries \mathcal{A} there is a negligible function $\text{negl}(\cdot)$ such that,*

$$\Pr \left[\text{Vf}_{\text{pk}}(m^*, \sigma^*) = 1 \wedge \left(\begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}_{\text{sk}}(\cdot)}(\text{pk}) \end{array} \right) \notin \mathcal{Q}^{\text{Sign}} \right] \leq \text{negl}(\lambda).$$

where the environment keeps track of the queries to and from the signing oracle via $\mathcal{Q}^{\text{Sign}}$.

2.4 The Permuted Kernel Problem and its Variants

In this subsection we give definitions of the computationally hard problems underlying the security of our proposed signature scheme.

We start by defining the classical Permuted Kernel Problem [51] in its generic form, similar to [50]: with an inhomogeneous syndrome \mathbf{y} as well as a dimension parameter t .

Definition 2.8 (IPKP problem). *Let (q, m, n, t) be positive integers such that $m < n$, $\mathbf{H} \in \mathbb{F}_q^{m \times n}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ and $\pi \in \mathcal{S}_n$ be a permutation such that $\mathbf{H}(\pi[\mathbf{x}_i]) = \mathbf{y}_i$ for $i \in [t]$. Furthermore, the matrix whose columns are the \mathbf{x}_i has full rank. Given $(\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]})$, the Inhomogeneous Permuted Kernel Problem $\text{IPKP}(q, m, n, t)$ asks to find π .*

The IPKP problem was originally introduced with $t = 1$ only; in the rest of the article we refer to this version of the problem as *mono-dimensional* IPKP. Correspondingly, we refer with *multi-dimensional* IPKP to instances with arbitrary choices of $t > 1$.

Instead of directly relying on the hardness of IPKP, we consider a relaxed version r -IPKP which allows for more efficient constructions. In this relaxed variant the searched permutation does not necessarily have to satisfy the identity for all given pairs but only for an arbitrary (non-zero) linear combination of those pairs.

Definition 2.9 (r-IPKP). Let (q, m, n, t) be positive integers such that $m < n$, $\mathbf{H} \in \mathbb{F}_q^{m \times n}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ and $\pi \in \mathcal{S}_n$ be a permutation such that $\mathbf{H}(\pi[\mathbf{x}_i]) = \mathbf{y}_i$ for $i \in [t]$. Furthermore, the matrix whose columns are the \mathbf{x}_i has full rank. Given $(\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]})$, the Relaxed Inhomogeneous Permuted Kernel Problem r -IPKP(q, m, n, t) asks to find any $\tilde{\pi} \in \mathcal{S}_n$ such that $\mathbf{H} \left(\tilde{\pi} \left[\sum_{i \in [t]} \kappa_i \cdot \mathbf{x}_i \right] \right) = \sum_{i \in [t]} \kappa_i \cdot \mathbf{y}_i$ for any $\kappa \in \mathbb{F}_q^t \setminus \mathbf{0}$, where $\kappa := \{\kappa_1, \dots, \kappa_t\}$ and $\mathbf{0} \in \mathbb{F}_q^t$ is the all zero vector.

The respective hardness of the two above problems are discussed in Section 4.3.

3 PoK and Signature based on r -IPKP

Recall that our goal is to construct a post-quantum digital signature scheme based on the ZKPoK protocol by using the Fiat-Shamir transformation [26]. However, Kales and Zaverucha in [36] showed that 5-round PoK which use parallel repetition to achieve a negligible soundness error can be attacked when they are converted to their non-interactive version with the Fiat-Shamir transform. The attack strategy optimally guesses the challenges in each round allowing to convince the verifier by crafting the responses based on the guessed challenge values, without actually knowing the secret. While this attack can be thwarted by increasing the number of parallel repetitions appropriately, it was shown in [15] that one can achieve a similar result by *amplifying the challenge space* queried by the verifier, which can lead to more efficient constructions.

Our construction leverage the r -IPKP problem introduced in Section 2.4 as a way to perform challenge space amplification. Informally, the problem requires, given a matrix \mathbf{H} and t pairs of vectors $(\mathbf{x}_i, \mathbf{y}_i)$, to find a permutation π that sends $\mathbf{H}\pi(\mathbf{x})$ to \mathbf{y} where $\mathbf{x} := \sum_i \kappa_i \mathbf{x}_i$ (resp. $\mathbf{y} := \sum_i \kappa_i \mathbf{y}_i$) and $\kappa_1, \dots, \kappa_t$ are the coefficients of an arbitrary adversarially chosen (non-zero) linear combination. Let $x = (\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]})$ and let $w = \pi \in \mathcal{S}_n$ as defined in Definition 2.9. Let $\mathcal{R}_{t-r\text{-IPKP}}$ be a relation for r -IPKP problem defined as,

$$\mathcal{R}_{t-r\text{-IPKP}} := \left\{ \left((\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]}) ; \tilde{\pi} \right) : \begin{array}{l} \mathbf{H} \left(\tilde{\pi} \left[\sum_{i \in [1,t]} \kappa_i \cdot \mathbf{x}_i \right] \right) = \sum_{i \in [1,t]} \kappa_i \cdot \mathbf{y}_i \\ \text{for any } \kappa \in \mathbb{F}_q^t \setminus \mathbf{0} \end{array} \right\}$$

We now present our protocol in Fig. 1 that is inspired from [15] and [24]. Informally, it consists of three main steps, following the MPCitH paradigm:

1. In the commitment step, the witness π is split into N *compositional* shares π_1, \dots, π_N such that $\pi = \pi_N \circ \pi_{N-1} \circ \dots \circ \pi_1$. The prover also generates

- N (pseudo) random vectors $\mathbf{v}_1, \dots, \mathbf{v}_N$ in \mathbb{F}_q^n . The compositional and vector shares are then combined to construct a syndrome $\mathbf{H}\mathbf{v}$ (the vector \mathbf{v} is generated by combining the shares π_i s and \mathbf{v}_i s, refer Fig. 1 for the details), which is committed together with the generated shares (π_i and \mathbf{v}_i).
2. The verifier then sends coefficients κ_i of an \mathbb{F}_q -linear combination as a first challenge. The prover then computes values $\mathbf{s}_1, \dots, \mathbf{s}_N$ with the help of the π_i and \mathbf{v}_i values committed earlier and the public statement $x = (\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]})$, such that $\mathbf{H}\mathbf{s}_N = \mathbf{H}\mathbf{v} + \sum_{i \in [t]} \kappa_i \mathbf{y}_i$. The prover then sends \mathbf{s}_i values as its response. In the actual protocol we use a collision-resistant hash function to compress the information sent to the verifier.
 3. Finally, the verifier sends an index $\alpha \in [N]$ as the second challenge. The prover reveals all shares π_i and \mathbf{v}_i except the ones with index α . Additionally, the prover reveals the share \mathbf{s}_α . This allows the verifier to verify the consistency of the views of all the shares except the ones with index α by recomputing the commitments. The verifier can also recompute all the \mathbf{s}_i values for $i \neq \alpha$ and together with \mathbf{s}_α sent by the prover, the verifier can then reconstruct \mathbf{s}_N . Finally the verifier computes $\mathbf{H}\mathbf{v} = \mathbf{H}\mathbf{s}_N - \sum_{i \in [t]} \kappa_i \mathbf{y}_i$ and checks if this value is consistent with the commitment received in first message (Step 1 above).

Theorem 3.1 (Completeness). *The protocol presented in Fig. 1 is perfectly complete.*

Proof. The completeness follows from the protocol description once it is observed that $\mathbf{s}_N = \pi \left[\sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i \right] + \mathbf{v}$ which implies that

$$\mathbf{H}\mathbf{s}_N - \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{y}_i = \mathbf{H} \left(\pi \left[\sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i \right] \right) + \mathbf{H}\mathbf{v} - \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{y}_i = \mathbf{H}\mathbf{v}.$$

Therefore for every true statement $(\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$ with witness π if the protocol described in Fig. 1 is executed honestly then the verifier \mathbf{V} accepts with probability 1 for all possible random choices of \mathbf{P} and \mathbf{V} . \square

Theorem 3.2 (Knowledge Soundness). *The protocol presented in Fig. 1 is knowledge sound with knowledge error*

$$\epsilon_{\text{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}.$$

Theorem 3.3 (Special Honest-Verifier Zero Knowledge). *Assume that there exists a $(t, \epsilon_{\text{PRG}})$ -secure PRG, and the commitment scheme Com is $(t, \epsilon_{\text{Com}})$ -hiding. Then there exists an efficient simulator Sim which, outputs a transcript such that no distinguisher running in time at most $t(\lambda)$ can distinguish between the transcript produced by Sim and a real transcript obtained by honest execution of the protocol in Fig. 1 with probability better than $(\epsilon_{\text{PRG}}(\lambda) + \epsilon_{\text{Com}}(\lambda))$.*

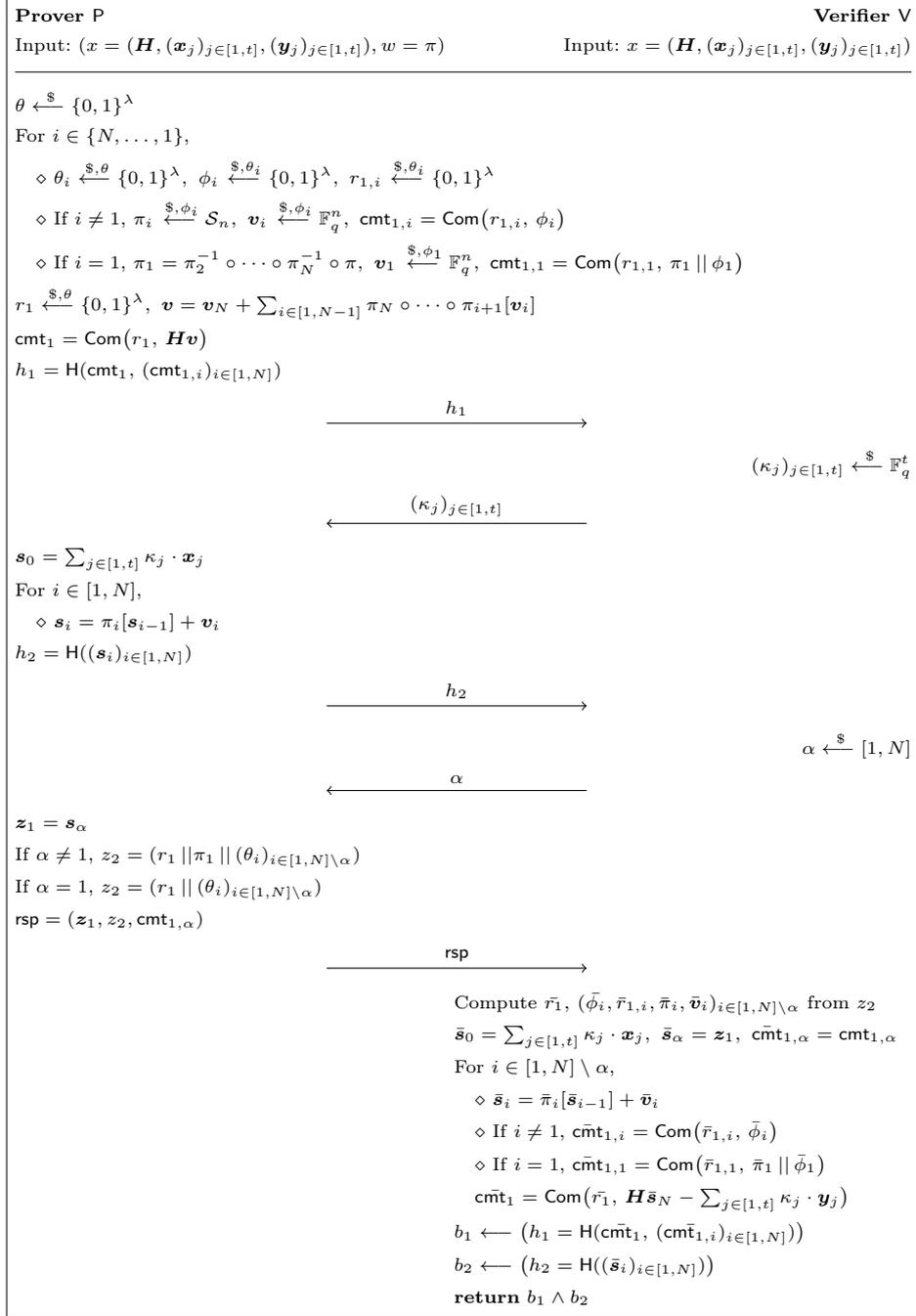


Fig. 1: PoK leveraging structure for the r-IPKP problem

We prove Theorem 3.2 and Theorem 3.3 in Section B.1 and Section B.2 of the supplementary material respectively. By applying the Fiat-Shamir transformation on the protocol shown in Fig. 1, one gets the digital signature scheme described in Fig. 2, Fig. 3 and Fig. 4.

1. Sample $\text{sk_seed} \xleftarrow{\$} \{0, 1\}^\lambda$ and $\text{pk_seed} \xleftarrow{\$} \{0, 1\}^\lambda$
2. Sample $\pi \leftarrow \text{PRG}(\text{sk_seed})$ from \mathcal{S}_n
3. Sample $(\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}) \leftarrow \text{PRG}(\text{pk_seed})$ from $\mathbb{F}_q^{m \times n} \times (\mathbb{F}_q^n)^t$
3. For $j \in [1, t]$,
 - ◊ Compute $\mathbf{y}_j = \mathbf{H}\pi[\mathbf{x}_j]$
4. Output $(\text{sk}, \text{pk}) = (\text{sk_seed}, (\text{pk_seed}, (\mathbf{y}_j)_{j \in [1, t]}))$

Fig. 2: PERK - KeyGen algorithm

Theorem 3.4. *Suppose PRG is $(t, \epsilon_{\text{PRG}})$ -secure and any adversary running in time $t(\lambda)$ can solve the the underlying r-IPKP instance with probability at most $\epsilon_{\text{r-IPKP}}$. Model \mathbf{H}_0 , \mathbf{H}_1 , and \mathbf{H}_2 as random oracles where \mathbf{H}_0 , \mathbf{H}_1 , and \mathbf{H}_2 have 2λ -bit output length. Then chosen-message attacker against the signature scheme (PERK) presented in Fig. 3, running in time $t(\lambda)$, making q_s signing queries, and making q_0 , q_1 , q_2 queries, respectively, to the random oracles, succeeds in outputting a valid forgery with probability*

$$\Pr[\text{Forge}] \leq \frac{(q_0 + \tau \cdot (N + 1) \cdot q_s)^2}{2 \cdot 2^{2\lambda}} + \frac{q_s \cdot (q_0 + q_1 + q_2 + q_s)}{2^{2\lambda}} \quad (3)$$

$$+ \tau \cdot q_s \cdot \epsilon_{\text{PRG}}(\lambda) + \epsilon_{\text{r-IPKP}} + q_2 \cdot \epsilon_{\text{KS}}^\tau,$$

where $\epsilon_{\text{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}$.

We prove Theorem 3.4 in Section C.1 of the supplementary material.

4 On the Hardness of r-IPKP

In this section we study the hardness of the r-IPKP from Definition 2.9. Even though, we introduce the r-IPKP together with our scheme its hardness is still tied to the hardness of the multi-dimensional and mono-dimensional versions of IPKP as we outline in the following.

Recall that the difference to IPKP is that for r-IPKP the solution does not necessarily have to be the permutation that works for all the given pairs, but it has to satisfy the PKP identity only for an arbitrary (non-zero) linear combination of those pairs, i.e. any permutation π such that

$$\mathbf{H} \left(\pi \left[\sum_i \kappa_i \mathbf{x}_i \right] \right) = \sum_i \kappa_i \mathbf{y}_i,$$

Inputs

- Secret key $\text{sk} = \pi$
- Public key $\text{pk} = (\mathbf{H}, (\mathbf{x}_j, \mathbf{y}_j)_{j \in [1, t]})$
- Message $m \in \{0, 1\}^*$

Step 1: Commitment

1. Sample salt and master seed $(\text{salt}, \text{mseed}) \xleftarrow{\$} \{0, 1\}^{2\lambda} \times \{0, 1\}^\lambda$
2. Sample seeds $(\theta^{(e)})_{e \in [1, \tau]} \leftarrow \text{PRG}(\text{salt}, \text{mseed})$ from $(\{0, 1\}^\lambda)^\tau$
3. For each iteration $e \in [1, \tau]$,
 - ◇ Compute $(\theta_i^{(e)})_{i \in [1, N]} \leftarrow \text{TreePRG}(\text{salt}, \theta^{(e)})$
 - ◇ For each party $i \in \{N, \dots, 1\}$,
 - If $i \neq 1$, sample $(\pi_i^{(e)}, \mathbf{v}_i^{(e)}) \leftarrow \text{PRG}(\text{salt}, \theta_i^{(e)})$ from $\mathcal{S}_n \times \mathbb{F}_q^n$
 - If $i = 1$, sample $\mathbf{v}_1^{(e)} \leftarrow \text{PRG}(\text{salt}, \theta_1^{(e)})$ from \mathbb{F}_q^n
 - If $i \neq 1$, compute $\text{cmt}_{1,i}^{(e)} = \text{H}_0(\text{salt}, e, i, \theta_i^{(e)})$
 - If $i = 1$, compute $\pi_1^{(e)} = (\pi_2^{(e)})^{-1} \circ \dots \circ (\pi_N^{(e)})^{-1} \circ \pi$ and $\text{cmt}_{1,1}^{(e)} = \text{H}_0(\text{salt}, e, 1, \pi_1^{(e)}, \theta_1^{(e)})$
 - ◇ Compute $\mathbf{v}^{(e)} = \mathbf{v}_N^{(e)} + \sum_{i \in [1, N-1]} \pi_N^{(e)} \circ \dots \circ \pi_{i+1}^{(e)}[\mathbf{v}_i^{(e)}]$ and $\text{cmt}_1^{(e)} = \text{H}_0(\text{salt}, e, \mathbf{H}\mathbf{v}^{(e)})$

Step 2: First Challenge

4. Compute $h_1 = \text{H}_1(\text{salt}, m, \text{pk}, (\text{cmt}_1^{(e)}, \text{cmt}_{1,i}^{(e)})_{e \in [1, \tau], i \in [1, N]})$
5. Sample $(\kappa_j^{(e)})_{e \in [1, \tau], j \in [1, t]} \leftarrow \text{PRG}(h_1)$ from $(\mathbb{F}_q^t)^\tau$

Step 3: First Response

6. For each iteration $e \in [1, \tau]$,
 - ◇ Compute $\mathbf{s}_0^{(e)} = \sum_{j \in [1, t]} \kappa_j^{(e)} \cdot \mathbf{x}_j$
 - ◇ For each party $i \in [1, N]$,
 - Compute $\mathbf{s}_i^{(e)} = \pi_i^{(e)}[\mathbf{s}_{i-1}^{(e)}] + \mathbf{v}_i^{(e)}$

Step 4: Second Challenge

7. Compute $h_2 = \text{H}_2(\text{salt}, m, \text{pk}, h_1, (\mathbf{s}_i^{(e)})_{e \in [1, \tau], i \in [1, N]})$
8. Sample $(\alpha^{(e)})_{e \in [1, \tau]} \leftarrow \text{PRG}(h_2)$ from $([1, N])^\tau$

Step 5: Second Response

9. For each iteration $e \in [1, \tau]$,
 - ◇ Compute $\mathbf{z}_1^{(e)} = \mathbf{s}_\alpha^{(e)}$
 - ◇ If $\alpha^{(e)} \neq 1$, $\mathbf{z}_2^{(e)} = (\pi_1^{(e)} \parallel (\theta_i^{(e)})_{i \in [1, N] \setminus \alpha^{(e)}})$
 - ◇ If $\alpha^{(e)} = 1$, $\mathbf{z}_2^{(e)} = (\theta_i^{(e)})_{i \in [1, N] \setminus \alpha^{(e)}}$
 - ◇ Compute $\text{rsp}^{(e)} = (\mathbf{z}_1^{(e)}, \mathbf{z}_2^{(e)}, \text{cmt}_{1, \alpha^{(e)}}^{(e)})$
10. Compute $\sigma = (\text{salt}, h_1, h_2, (\text{rsp}^{(e)})_{e \in [1, \tau]})$

Fig. 3: PERK - Sign algorithm

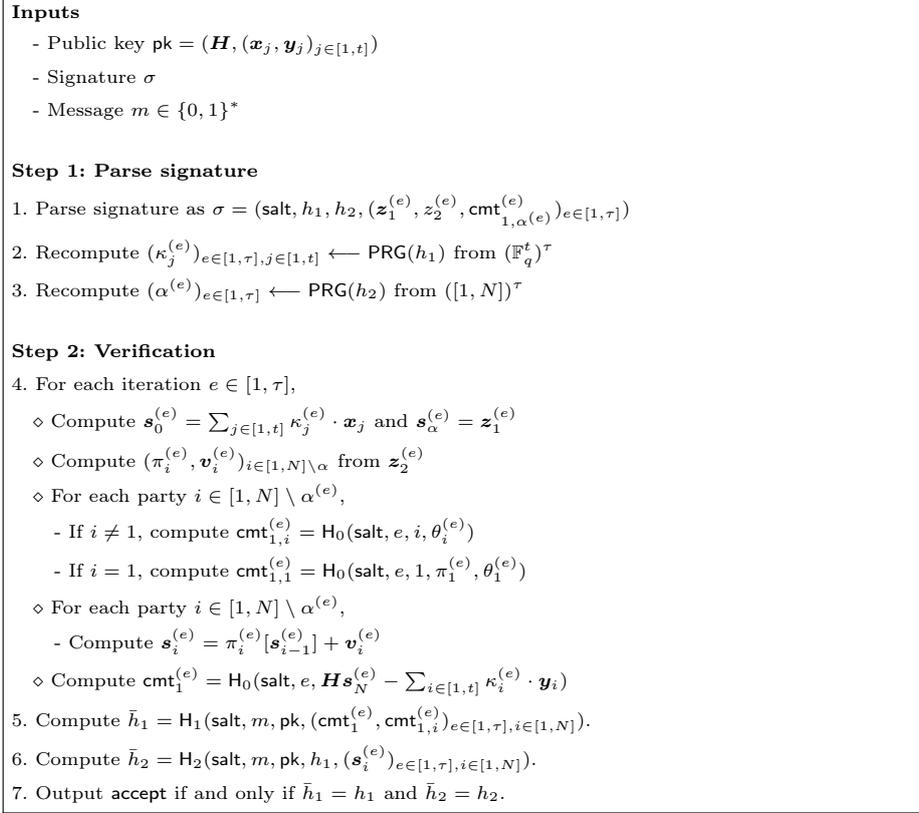


Fig. 4: PERK - Verify algorithm

for a choice of the $\kappa_i \in \mathbb{F}_q$, is a solution. Clearly any algorithm applicable to IPKP can also be applied to find a solution to the r-IPKP problem. However, a solution to r-IPKP does not necessarily have to be a solution to IPKP. Therefore algorithms to solve r-IPKP can be split into those initially proposed for IPKP and those specifically designed to solve r-IPKP. In the following section we first describe known approaches for solving IPKP, after which we present a new algorithm in Section 4.2, Algorithm 1 specifically designed to solve r-IPKP.

4.1 Attacks on IPKP

The IPKP problem was introduced by Shamir in 1990 [51]. Still, the best attack on mono-dimensional IPKP is a meet-in-the-middle adaptation known as the KMP algorithm by Koussa, Macario-Rat and Patarin [38]. This algorithm extends easily to $t > 1$, which was recently formalized in [50]. The multi-dimensional IPKP first appeared in the literature in 2011 [39]. However, until recently cryptanalysis only resulted in better algorithms for the particular case of binary fields [44]. Recently, Santini, Baldi and Chiraluce [50] proposed the

SBC algorithm which extends the KMP algorithm by a pre-processing step. For $t > 1$, i.e., for the multi-dimensional case, this results in improvements over the KMP approach. In the following we give a brief overview of those attacks, as well as a simple observation that when $t \geq n$, the IPKP becomes polynomial. For fully-fledged descriptions, analysis and estimation scripts the reader is referred to [20, 38, 50].

The KMP Algorithm. The algorithm by Koussa, Macario-Rat and Patarin [38] is a slight variant of previously known combinatorial techniques [8, 30, 35, 45]. Here we outline first the initial proposal for the mono-dimensional IPKP [38].

Initially, the matrix \mathbf{H} is transformed into semi-systematic form by applying a change of basis (modelled by the invertible matrix \mathbf{Q})

$$\mathbf{QH} = \begin{pmatrix} \mathbf{I}_{m-u} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix},$$

where $\mathbf{H}_1 \in \mathbb{F}_q^{(m-u) \times (n-m+u)}$, $\mathbf{H}_2 \in \mathbb{F}_q^{u \times (n-m+u)}$ and u is an optimization parameter of the algorithm. By multiplying the syndrome \mathbf{y} by the same matrix \mathbf{Q} one maintains the validity of the PKP identity

$$\begin{aligned} \mathbf{QH}\pi(\mathbf{x}) &= \begin{pmatrix} \mathbf{I}_{m-u} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix} \pi(\mathbf{x}) = \begin{pmatrix} \mathbf{I}_{m-u} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = (\mathbf{x}_1 + \mathbf{H}_1\mathbf{x}_2, \mathbf{H}_2\mathbf{x}_2)^\top \\ &= (\mathbf{y}_1, \mathbf{y}_2)^\top = \mathbf{Q}\mathbf{y}, \end{aligned}$$

where $\mathbf{Q}\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{F}_q^{m-u} \times \mathbb{F}_q^u$ and $\pi(\mathbf{x}) = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{F}_q^{m-u} \times \mathbb{F}_q^u$. The algorithm now focuses on solving the identity $\mathbf{H}_2\mathbf{x}_2 = \mathbf{y}_2$. For any found \mathbf{x}_2 satisfying the identity it is then checked if $\mathbf{x}_1 = \mathbf{y}_1 - \mathbf{H}_1\mathbf{x}_2$ and \mathbf{x}_2 together form a permutation of \mathbf{x} .

Candidates for \mathbf{x}_2 are obtained by a meet-in-the-middle enumeration strategy. Therefore \mathbf{x}_2 is further split as $\mathbf{x}_2 = (\mathbf{x}_{21}, \mathbf{x}_{22})$, with $\mathbf{x}_{21}, \mathbf{x}_{22} \in \mathbb{F}_q^{u \times ((n-m+u)/2)}$ to obtain the meet-in-the-middle identity

$$\mathbf{H}_2(\mathbf{x}_{21}, \mathbf{0}) = \mathbf{y}_2 - \mathbf{H}_2(\mathbf{0}, \mathbf{x}_{22}). \quad (4)$$

Then the algorithm enumerates all candidates for \mathbf{x}_{21} and \mathbf{x}_{22} , that is all permutations of any selection of $(n-m+u)/2$ entries of \mathbf{x} . For each such vector the left (resp. right) side of Eq. (4) is stored in a list L_1 (resp. L_2). In a final step the algorithm searches for matches between the lists L_1 and L_2 yielding the candidates for \mathbf{x}_2 . From there \mathbf{x}_1 can be computed as $\mathbf{x}_1 = \mathbf{y}_1 - \mathbf{H}_1\mathbf{x}_2$. If $(\mathbf{x}_1, \mathbf{x}_2)$ forms a permutation of \mathbf{x} this yields the solution π .

The complexity of the algorithm is (up to polynomial factors) linear in the sizes of the lists L_1 , L_2 and L , where L is the list of matches. The expected sizes are

$$|L_1| = |L_2| = \binom{n}{(n-m+u)/2} ((n-m+u)/2)! \quad \text{and} \quad |L| = \frac{|L_1 \times L_2|}{q^u}$$

Extension to multi-dimensional IPKP. The algorithm can easily be extended to solve IPKP for arbitrary t , as it was recently made explicit in [50]. Therefore let \mathbf{X} be the matrix containing the \mathbf{x}_i as rows and \mathbf{Y} containing the \mathbf{y}_i as columns. Substituting the occurrences of \mathbf{x} and \mathbf{y} by \mathbf{X} and \mathbf{Y} resp., where the permutation now operates as a column permutation on matrices, one obtains this generalization. Then of course the definition of $(\mathbf{x}_1, \mathbf{x}_2)$ and $(\mathbf{y}_1, \mathbf{y}_2)$ analogously extends to matrices.

In terms of complexity, the enumeration effort stays (up to polynomial factors) exactly the same, as the possible number of permutations remains unchanged. The only difference is that the expected size of the list L of matches reduces to $\frac{|L_1 \times L_2|}{q^{u \cdot t}}$.

The SBC algorithm. The algorithmic improvement by Santini-Baldi-Chiraluce (SBC) extends the KMP algorithm by a preprocessing step.

Therefore, assume that the matrix \mathbf{H}_2 constructed in the KMP algorithm would contain zero columns. Clearly, those columns could be removed without affecting the validity of the identity $\mathbf{H}_2 \mathbf{x}_2 = \mathbf{y}_2$. But in turn this would reduce the enumeration effort to find candidates for \mathbf{x}_2 .

The preprocessing step of the SBC algorithm now consists in finding a u -dimensional subcode of \mathbf{H} that has small support $w < n - m + u$, i.e., an \mathbf{H}_2 that contains some zero columns. This can be accomplished by adaptations of Information Set Decoding (ISD) algorithms [11]. Subsequently, the SBC algorithm continues as the KMP algorithm by finding candidates for \mathbf{x}_2 in $\mathbf{H}_2 \mathbf{x}_2 = \mathbf{y}_2$, now with reduced enumeration complexity. This resulting list of candidates is now treated as the first list, L_1 , in the KMP algorithm.

Note that the KMP algorithm creates two lists each giving candidates for $(n - m + u)/2$ entries of the permutation. Now, as there are already candidates for w entries in L_1 , the second list enumerates the permutation for further $n - m + u - w$ positions. Eventually both lists are matched on $u \cdot t$ coordinates as in the KMP algorithm to obtain a list of final candidates. Note that as in the KMP algorithm now each candidate of the final list reveals $n - m + u$ potential positions of the permutation which can be checked in polynomial time for extending to a full solution.

A polynomial attack when $t \geq n$. We describe in this paragraph a simple polynomial attack on the inhomogeneous IPKP problem when $t \geq n$. In that case, the vectors $(\mathbf{x}_1, \dots, \mathbf{x}_t)$ from the IPKP instance generate the whole \mathbb{F}_q^n , since by definition the matrix whose columns are the \mathbf{x}_i has full rank. For any index $1 \leq i \leq n$, there exists a linear combination such that $\sum_j \lambda_j \mathbf{x}_j = (\dots, 0, 1, 0, \dots)$, the unique non-zero coordinate being taken at index i . We can then obtain a vector by taking the same linear combination on the \mathbf{y}_j : $\mathbf{y} = \sum_j \lambda_j \mathbf{y}_j$. If there exists a solution π to the given instance, then \mathbf{y} is bound to be equal to a column of \mathbf{H} whose index yields $\pi(i)$. The attacker can then reconstruct the solution permutation π in linear time $\mathcal{O}(n)$. This attack does not apply to r-IPKP because in that case, the solution π is non necessarily a solution to the mono-dimensional instance involving the linear combinations $\mathbf{x} = \sum_j \lambda_j \mathbf{x}_j = (\dots, 0, 1, 0, \dots)$ and

$\mathbf{y} = \sum_j \lambda_j \mathbf{y}_j$. This indicates that both problems are fundamentally different and that a specific strategy should be devised for solving r-IPKP.

4.2 A new Algorithm Solving r-IPKP

We introduce the r-IPKP problem together with our scheme. However, it is still very related to the multi-dimensional and mono-dimensional versions of IPKP and their corresponding hardness. We already discussed the relation to the *multi-dimensional* case of IPKP. Let us now focus on the relation between r-IPKP and the *mono-dimensional* version of IPKP. In this context r-IPKP can be seen as a multi-instance version of IPKP.

Therefore disregard the (most likely) unique solution to r-IPKP which simultaneously solves IPKP for the same t , i.e. the permutation that works for all pairs $(\mathbf{x}_i, \mathbf{y}_i)$. Further, assume that for any of the given pairs $(\mathbf{x}_i, \mathbf{y}_i)$ there exist a permutation π_i solving the corresponding mono-dimensional IPKP instance, that is a π_i that satisfies $\mathbf{H}\pi_i[\mathbf{x}_i] = \mathbf{y}_i$. If we would be forced to recover one of the π_i , this would exactly be a multi-instance version of IPKP. However, the r-IPKP allows to recover not only those but also any permutation that works for an arbitrary linear combination of the given pairs. Clearly, this gives a total of q^t different pairs, but unlike the multi-instance case those pairs are related.

In fact, from a coding theory perspective the r-IPKP asks to recover a permutation that works for any codeword and the corresponding syndrome where the code is defined by the generator matrix containing the \mathbf{x}_i s as rows. In the following we give a new algorithm for solving r-IPKP that exploits this view on the problem. The algorithm is based on a preprocessing of the given pairs $(\mathbf{x}_i, \mathbf{y}_i)$, a subsequent instance permutation and an adapted KMP-style enumeration technique. Moreover, our algorithm contains the KMP algorithm as a special case.

The algorithm starts by finding a low Hamming-weight codeword in the code whose generator matrix contains the \mathbf{x}_i as rows. This task is accomplished by application of an ISD algorithm. Let $\mathbf{x}' = \sum \kappa_i \mathbf{x}_i$ be this codeword of weight w and $\mathbf{y}' = \sum \kappa_i \mathbf{y}_i$ the corresponding syndrome.

We now focus on finding a permutation π that satisfies $\mathbf{H}(\pi[\mathbf{x}']) = \mathbf{y}'$. Therefore, we apply a KMP-style enumeration with some modifications. Again we derive the identity $\mathbf{H}_2 \mathbf{x}_2 = \mathbf{y}_2 \in \mathbb{F}_q^u$, with $\mathbf{x}_2 = (\mathbf{x}_{21}, \mathbf{x}_{22})$ as in the usual KMP algorithm. Now, recall that $\pi[\mathbf{x}'] = (\mathbf{x}_1, \mathbf{x}_2)$ contains $n - w$ zeros. For the enumeration of \mathbf{x}_{21} and \mathbf{x}_{22} we now assume that z of those zeros are mapped into \mathbf{x}_2 by the permutation. Moreover, we assume that $z/2$ of those zeros are mapped to \mathbf{x}_{21} and $z/2$ to \mathbf{x}_{22} . This leads to a reduced amount of candidates for $\mathbf{x}_{21}, \mathbf{x}_{22}$ that has to be enumerated.

Of course we do not know a priori if the permutation indeed distributes $z/2$ zeros onto \mathbf{x}_{21} and $z/2$ zeros onto \mathbf{x}_{22} . Therefore prior to the enumeration we apply a random permutation to the columns of \mathbf{H} to redistribute the weight (and zeros) of $\pi[\mathbf{x}']$. If the enumeration does not lead to a solution we repeat with a different column permutation of \mathbf{H} .

A pseudocode description is given in Algorithm 1. Note that for $w = n$, i.e., a maximum-weight codeword, we resemble the standard KMP algorithm for solving mono-dimensional IPKP.

Algorithm 1: Algorithm Solving r -IPKP

Input : r -IPKP instance $(\mathbf{H}, (\mathbf{x}_i, \mathbf{y}_i)_{i \in [t]})$
Output: solution $\pi, \kappa_i \in \mathbb{F}_q, i \in [t]$

- 1 For a vector \mathbf{x} and an integer k let $\mathcal{P}_{\mathbf{x},k}$ be the set of vectors of length k with entries from \mathbf{x} (with their maximum occurrence as in \mathbf{x}).
- 2 Choose optimal positive $u \leq n - m$, $w \leq n$, and $z \leq n - w$, let
 $k := (n - m + u)/2$
- 3 Find weight- w codeword $\mathbf{x}' = \sum_i \kappa_i \mathbf{x}_i$ in the code defined by the \mathbf{x}_i
- 4 Let $\mathbf{y}' = \sum_i \kappa_i \mathbf{y}_i$
- 5 **repeat**
- 6 choose random permutation π'
- 7 $\mathbf{H}^* = \pi'[\mathbf{H}]$
- 8 $\mathbf{H}' = \mathbf{Q}\mathbf{H}^* = \begin{pmatrix} \mathbf{I}_{m-u} & \mathbf{H}_1 \\ \mathbf{0} & \mathbf{H}_2 \end{pmatrix}, (\mathbf{y}_1, \mathbf{y}_2) = \mathbf{Q}\mathbf{y}'$
- 9 $L_1 = \{(H_2(\mathbf{z}_1, 0^k), \mathbf{z}_1) \mid \mathbf{z}_1 \in \mathcal{P}_{\mathbf{x},k} \wedge \text{wt}(\mathbf{z}_1) = k - z/2\}$
- 10 $L_2 = \{(\mathbf{y}_2 - H_2(0^k, \mathbf{z}_2), \mathbf{z}_2) \mid \mathbf{z}_2 \in \mathcal{P}_{\mathbf{x},k} \wedge \text{wt}(\mathbf{z}_2) = k - z/2\}$
- 11 Compute $L = \{(\mathbf{z}_1, \mathbf{z}_2) \in L_1 \times L_2 \mid \mathbf{H}_2(\mathbf{z}_1, \mathbf{z}_2) = \mathbf{y}_2\}$ from L_1, L_2
- 12 **foreach** $\mathbf{x}_2 \in L$ **do**
- 13 $\mathbf{x}_1 = \mathbf{y}_1 - \mathbf{H}_1 \mathbf{x}_2$
- 14 **if** $\exists \tilde{\pi}: \tilde{\pi}[\mathbf{x}_1, \mathbf{x}_2] = \mathbf{x}'$ **then**
- 15 **return** $(\pi')^{-1} \circ \tilde{\pi}$

Analysis of Algorithm 1. Let us start with the correctness of the algorithm.

Correctness. Note that the permuted instance $(\mathbf{H}^*, \mathbf{x}', \mathbf{y}')$ with $\mathbf{H}^* = \pi'[\mathbf{H}]$ has solution $\pi' \circ \pi$ if π solves the original instance $(\mathbf{H}, \mathbf{x}', \mathbf{y}')$. Therefore the algorithm correctly returns $(\pi')^{-1} \circ \tilde{\pi}$ as the solution to the original instance, where $\tilde{\pi}$ solves the permuted instance.

Accordingly the solution to the permuted instance is $(\mathbf{x}_1, \mathbf{x}_2) = \pi'[\pi[\mathbf{x}']]$. Line 9 to 11 enumerate all candidates for \mathbf{x}_2 satisfying $\mathbf{H}_2 \mathbf{x}_2 = \mathbf{y}_2$, where $\mathbf{x}_2 = (\mathbf{z}_1, \mathbf{z}_2)$ with each of the \mathbf{z}_i containing $z/2$ zeros. Note that by construction $(\mathbf{x}_1, \mathbf{x}_2)$ contains $n - w$ zeros. As the permutation π' redistributes the zeros the algorithm can recover the permutation.

Complexity. The complexity of the algorithm splits into the cost of finding the short codeword \mathbf{x}' and the cost of the **repeat** loop. The codeword is found by application of an ISD algorithm. Let us denote this cost by T_{ISD} .

The cost of the loop is equal to the amount of repetitions times the cost of one iteration. The amount of different permutations until the zeros are distributed as desired is

$$P = \frac{\binom{n}{n-w}}{\binom{n-2k}{n-w-z} \binom{k}{z/2}^2},$$

where $k = (n - m + u)/2$. The cost for one iteration is (up to polynomial factors) linear in the involved lists' sizes. Note that we have

$$|L_i| = \binom{k}{z/2} \binom{n-z}{k-z/2} (k-z/2)! \quad \text{and} \quad |L| = \frac{|L_1 \times L_2|}{q^u}.$$

The total time complexity therefore amounts to

$$T = \tilde{O}(T_{\text{ISD}} + (|L_1| + |L|) \cdot P),$$

while the memory complexity is equal to $M = \tilde{O}(|L_1| + |L|)$.

In our numerical optimization we use for T_{ISD} the basic ISD procedure by Prange [47] which gives

$$T_{\text{ISD}} = \tilde{O}\left(\frac{\binom{n}{w}}{\binom{n-m}{w}}\right).$$

There are more sophisticated ISD procedures with lower cost, but as T_{ISD} does not dominate the running time, we refrain from further optimizations.

Note that this running time assumes a single existing solution for the considered mono-dimensional instance solved. In case of multiple solutions the running time can be lower, which we discuss in the next section.

Further improvement of Algorithm 1. For completeness, we point out that the algorithm can be (slightly) improved by considering in L_1 and L_2 vectors of weight $k - z \leq \text{wt}(\mathbf{z}_i) \leq k - z/2$. This would only insignificantly increase the list sizes, while giving a slightly higher probability of the permutation distributing the weight as desired. However, the overall improvement is a factor of order roughly 2. For small values of $t \leq 5$ as considered later, the factor turns out to be smaller than $\sqrt{2}$. Therefore for the sake of clarity we omit this improvement.

Another intuitive strategy seems the consideration of small support subcodes instead of small codewords in the code defined by the \mathbf{x}_i . This would lead to a reduced amount of matches, i.e., reduced list size L at the expense of larger w and correspondingly smaller z . However, the nature of instances considered in PERK renders this strategy ineffective. As detailed later, instances are chosen, such that for any single codeword there exist exponentially many solutions, leading to an exponential speedup. On the other hand already when considering a subcode of dimension two, the only existing solution is the permutation solving the IPKP

defined by all pairs $(\mathbf{x}_i, \mathbf{y}_i)$. We find that the reduced amount of matches from considering subcodes does not compensate for the speedup from the amount of solutions. We therefore again omit further details for the sake of clarity.

4.3 Concrete Complexity of Solving r-IPKP

In this section we give details on how the variation of different parameters affects the hardness of r-IPKP. A solid understanding of those effects is crucial for secure parameter selection. As outlined previously, for solving the r-IPKP one can either directly apply an IPKP algorithm (see Section 4.1) or solve one of the single IPKP instances defined by any linear combination of input pairs (see Section 4.2). Which of the two attack strategies is more efficient depends on the particular choice of parameters.

Effect of the number of solutions. Multiple existing solutions can lead to a maximum speedup that is linear in this amount of solutions. Whether this maximum speedup can be realized depends on the particular algorithm. However, for our parameter selection we conservatively assume that any algorithm can leverage this maximum speedup.

Note that the expected number of solutions differs for the considered sub-problems. The expected number of solutions for any random IPKP instance is about $\text{Sol}_{\text{IPKP}} = \frac{n!}{q^{m-t}}$. Note that the *mono*-dimensional IPKP instance solved in the context of our new Algorithm 1 is not random but contains z zeros. In this case the amount of expected solutions is only $\text{Sol}_{\text{IPKP},z}^{\text{mono}} = \frac{n!}{q^m z!}$.

Effect of t on the running time. Santini et al. [50] observed that the running time of the KMP algorithm as well as the running time of their SBC algorithm for IPKP is asymptotically independent of t .³ For concrete parameters, these algorithms still yield speedups for increasing t but the running time quickly converges to a stable minimum. Therefore, based on known algorithms the hardness of IPKP does not seem to deteriorate for high values of t . Contrary, the complexity of our new Algorithm 1 is monotonically decreasing for increasing t . This shows that high choices of t in the r-IPKP context are vulnerable.

To visualize this we consider in Table 1 a fixed instance for increasing t . The SBC algorithm reaches its minimum running time already for $t = 10$, while Algorithm 1 constantly improves. However, the SBC algorithm has a lower complexity for small choices of t and obtains a larger gain for early increases.

Note that for the chosen parameters in Table 1 already a random mono-dimensional IPKP instance has at most one solution in expectation, i.e. $\text{Sol}_{\text{IPKP}}^{\text{mono}} \leq 1$.

³ Informally, this can be seen by observing that t only affects the amount of matches, i.e. the size of L (compare to Section 4.1). However, asymptotically the size of the initial lists L_i and L are balanced, therefore a decrease of L does not lead to runtime improvements.

t	T_{new}	T_{SBC}
1	139.35	139.35
2	139.01	116.25
3	137.46	110.70
10	115.98	88.18
15	100.27	88.18
19	89.14	88.18
20	85.63	88.18
25	71.77	88.18
30	63.47	88.18

Table 1: Complexity of SBC algorithm (T_{SBC}) and Algorithm 1 (T_{new}) for increasing t on r-IPKP(n, m, q, t) instance with $(n, m, q) = (66, 31, 1021)$.

Effect of m on the running time. Generally the hardness of IPKP is increasing with decreasing m . This holds up to the point where there exist multiple solutions. Previous parameter selection for PKP-based schemes therefore chooses m minimal such that there exists no more than one random solution in expectation. However, for the specific case of the r-IPKP problem, the two sub-problems, i.e., the multi- and mono-dimensional IPKP instances, have a different amount of expected solutions. Here, decreasing m leads, generally, only to an increase of the problem complexity as long as the solution to both sub-problems is still unique.

5 Parameters

In this section we present the parameters of our scheme. Generally the parameters divide into r-IPKP specific parameters, i.e., (q, n, m, t) , and MPC parameters, i.e., the number of parties N and the number of parallel repetitions τ . The rationale for their selection are as follows.

Selection of MPC parameters. The number of parties and iterations is governed by the knowledge soundness of the protocol. Following common practice we propose two different parameter sets, a *short* variant using $N = 256$ and a *fast* variant using $N = 32$. The number of protocol repetitions τ is then chosen to guarantee a soundness probability of $2^{-\lambda}$ for $\lambda \in \{128, 192, 256\}$ for category I, III and V respectively. For deriving the soundness we take into account the attack by Kales-Zavurecha, see Section D of the supplementary material.

Selection of problem parameters. For parameter selection we ensure that the complexity of the SBC algorithm as well as the complexity of our new Algorithm 1 are above the security threshold, when assuming a linear speedup from the existing amount of solutions. Note that it is important to consider both

strategies as the IPKP suggests to decrease m to increase the difficulty of the problem. This is related to the low amount of expected solutions Sol_{IPKP} , which allows to decrease m significantly without introducing multiple solutions. Contrary, for any mono-dimensional instance given by the possible linear combinations, there exist several solutions $\text{Sol}_{\text{IPKP}}^{\text{mono}}$ for such small choices of m , which decrease the complexity of Algorithm 1.

Note that we, conservatively, restrict in our parameter selection to small choices of $t \in \{3, 5\}$ to guard against attacks that exploit the specifics of r-IPKP over IPKP. For such small values of t , the SBC algorithm has generally a lower complexity than Algorithm 1 (compare to Table 1). In those cases, the parameter selection process leads to a choice of m which implies a unique solution to the multi-dimensional IPKP instance, while there exist multiple solutions to the mono-dimensional instance solved in the context of Algorithm 1. This leads to a balancing of both complexities via the amount of solutions.

In our complexity estimations we ignore polynomial factors and ensure that the exponential factors of the complexity formulas already match the NIST security level definitions of 143, 207 and 272 bits of category I, III and V respectively. For the complexity estimation of the SBC algorithm we rely on the *CryptographicEstimators* library⁴ incorporating a more efficient version of the script from [50]. For our algorithm we rely on a separate script.

Overall this leads to the choices of parameters specified in Table 2.

Parameter Set	PKP parameters					MPC parameters			
	λ	q	n	m	t	N	τ	pk size	σ size
PERK-I-fast3	128	1021	79	35	3	32	30	0.15 kB	8.35 kB
PERK-I-fast5	128	1021	83	36	5	32	28	0.24 kB	8.03 kB
PERK-I-short3	128	1021	79	35	3	256	20	0.15 kB	6.56 kB
PERK-I-short5	128	1021	83	36	5	256	18	0.24 kB	6.06 kB
PERK-III-fast3	192	1021	112	54	3	32	46	0.23 kB	18.8 kB
PERK-III-fast5	192	1021	116	55	5	32	43	0.37 kB	18.0 kB
PERK-III-short3	192	1021	112	54	3	256	31	0.23 kB	15.0 kB
PERK-III-short5	192	1021	116	55	5	256	28	0.37 kB	13.8 kB
PERK-V-fast3	256	1021	146	75	3	32	61	0.31 kB	33.3 kB
PERK-V-fast5	256	1021	150	76	5	32	57	0.51 kB	31.7 kB
PERK-V-short3	256	1021	146	75	3	256	41	0.31 kB	26.4 kB
PERK-V-short5	256	1021	150	76	5	256	37	0.51 kB	24.2 kB

Table 2: Parameters of PERK signature scheme

Note that even though we are quite conservative in parameter selection by restricting to small choices of t , disregarding polynomial factors and assuming

⁴ https://github.com/Crypto-TII/cryptographic_estimators

a maximum speedup from multiple solutions, we obtain competitive signature sizes. Also all considered algorithms use as much memory as they consume time, which in a more realistic estimate that accounts for memory access leads to an even higher security margin.

In Table 3 we provide the corresponding time complexities in logarithmic scale of the SBC algorithm and Algorithm 1 (accounting for a linear speedup from multiple existing solutions) on the suggested parameter sets. As outlined, the parameter sets are chosen to balance both time complexities. Additionally the tables provide the internal parameters of the algorithms. In the case of Algorithm 1 we find that for small values of t as considered here, the choice $w = n - z$ is optimal, meaning all contained zeros should be contained in the part of the vector which is enumerated. For the SBC algorithm, recall that u is the dimension of the subcode and we denote by z the amount of zero columns in \mathbf{H}_2 (compare to Section 4.1), i.e., the subcode has support $w = n - m + u - z$.

Parameter Set	SBC [50]			Algorithm 1		
	time	u	z	time	u	z
PERK-I-fast3	145.7	5	1	147.5	17	2
PERK-I-fast5	147.7	3	2	147.2	19	0
PERK-III-fast3	210.7	7	1	210.1	26	2
PERK-III-fast5	212.5	4	2	210.8	27	4
PERK-V-fast3	274.8	9	1	275.5	35	4
PERK-V-fast5	274.1	6	2	275.5	37	3

Table 3: Parameters of PERK signature scheme

5.1 Key and Signature Sizes

Table 2 already states the public key and signature sizes for the different parameter sets. Let us give an overview how those numbers are composed.

Key size. The private key as well as most of the components of the public key can be derived from a seed. The only elements not generated from a seed in the public key are the t syndromes (\mathbf{y}_i) .

Signature size. The signature consists of a salt and two hashes (h_1, h_2) , making a subtotal of 6λ bits, and then τ repetitions of the following:

- A vector $\mathbf{z}_1^{(e)} \in \mathbb{F}_q^n$;
- A permutation in \mathcal{S}_n ;
- $N-1$ seeds (of size λ) arranged in a PRG tree, hence of size only $\lambda \cdot \lceil \log_2(N) \rceil$;

- A commitment $\text{cmt}_{1,\alpha^{(\epsilon)}}^{(\epsilon)}$ of size 2λ .

Overall, for a security level λ , the key and signature sizes for our signature scheme are captured by the following formulas:

Public key size (bits)
$\lambda + t \cdot m \lceil \log_2(q) \rceil$
Signature size (bits)
$6\lambda + \tau \cdot \left(\underbrace{n \lceil \log_2(q) \rceil}_{\text{vector in } \mathbb{F}_q^n} + \underbrace{n \lceil \log_2(n) \rceil}_{\text{permutation}} + \underbrace{\lambda \lceil \log_2(N) \rceil}_{\text{seeds}} + \underbrace{2\lambda}_{\text{commitment}} \right)$

Table 4: Public key and signature sizes in bits

Signature compression. Our implementation features an optimization that further reduce the aforementioned signature theoretical size. The idea is to pack the permutation two by two. Instead of representing a permutation $\pi \in \mathcal{S}_n$ as a sequence of n elements in $[0, n - 1]$ it is represented as a sequence of $\lceil n/2 \rceil$ elements in $[0, n^2 - 1]$. When the following inequality holds

$$\lceil \log_2(n^2) \rceil < 2 \lceil \log_2(n) \rceil,$$

the packed permutation takes less memory size. Numbers given in Table 2 take into account this compression technique.

5.2 Comparison

We compare our scheme with other signature schemes, either based on PKP or based on other security assumptions. The results are presented in Table 5 and Table 6.

6 Performances

This section provides performance measures of PERK signature. Our constant-time implementation is written in C, and uses AVX2 vector instructions. The benchmarks have been performed on a machine that has 64 GB of memory and an Intel[®] Core[™] i9-13900K @ 3.00 GHz for which the Hyper-Threading, Turbo Boost and SpeedStep features were disabled. For each parameter set, the results have been obtained by computing the average from 1000 random instances. The

Name	Variant	pk	σ	Security assumption
PKP-DSS [13]	-	0.1 kB	21.0 kB	PKP
SUSHYFISH [12]	fast	0.1 kB	18.4 kB	IPKP
	short	0.1 kB	12.1 kB	IPKP
BG22 [15]	fast	0.1 kB	10.0 kB	IPKP
	short	0.1 kB	8.9 kB	IPKP
Fen22 [21]	fast	0.1 kB	16.4 kB	IPKP
	short	0.1 kB	12.8 kB	IPKP
PERK-I-fast3	fast	0.15 kB	8.35 kB	r-IPKP
PERK-I-short5	short	0.24 kB	6.06 kB	r-IPKP

Table 5: Comparison of our scheme with other digital signature schemes based on PKP assumptions

Name	Variant	pk	σ	Security assumption
SPHINCS+ [32]	fast	0.03 kB	17.1 kB	Hash Collisions
	short	0.03 kB	7.9 kB	Hash Collisions
FJR22 [22]	fast	0.1 kB	9.7 kB	SD over \mathbb{F}_{256}
	short	0.1 kB	6.9 kB	SD over \mathbb{F}_{256}
Fen22 [21]	fast	0.1 kB	7.4 kB	RSD over \mathbb{F}_2
	short	0.1 kB	5.9 kB	RSD over \mathbb{F}_2
Fen22 [21]	fast	0.1 kB	7.2 kB	MinRank over \mathbb{F}_{16}
	short	0.1 kB	5.5 kB	MinRank over \mathbb{F}_{16}
Fen22 [21]	fast	0.1 kB	8.5 kB	\mathcal{MQ} over \mathbb{F}_{256}
	short	0.1 kB	7.1 kB	\mathcal{MQ} over \mathbb{F}_{256}
R-BG [7]	fast	0.1 kB	7.7 kB	Restricted-SD
	short	0.1 kB	7.2 kB	Restricted-SD
PERK-I-fast3	fast	0.15 kB	8.35 kB	r-IPKP
PERK-I-short5	short	0.24 kB	6.06 kB	r-IPKP

Table 6: Comparison of our scheme with other digital signature schemes not based on PKP assumptions

following optimization flags have been used during compilation: `-O3 -std=c99 -pedantic -funroll-all-loops -march=native -mavx2 -mpclmul -msse4.2 -maes`.

Parameter Set	Keygen	Sign	Verify
PERK-I-fast3	77 k	7.6 M	5.3 M
PERK-I-fast5	88 k	7.2 M	5.1 M
PERK-I-short3	80 k	39 M	27 M
PERK-I-short5	92 k	36 M	25 M
PERK-III-fast3	167 k	16 M	13 M
PERK-III-fast5	184 k	15 M	12 M
PERK-III-short3	174 k	82 M	65 M
PERK-III-short5	194 k	77 M	60 M
PERK-V-fast3	297 k	36 M	28 M
PERK-V-fast5	322 k	34 M	27 M
PERK-V-short3	299 k	184 M	142 M
PERK-V-short5	329 k	170 M	131 M

Table 7: Performances of our implementation for different instances of PERK. The key generation numbers are in kilo CPU cycles, while the signing and verification numbers are in million CPU cycles.

Data availability. Data sharing not applicable to this article as no datasets were generated or analysed during the current study. The reference implementation for the scheme is available at <https://pqc-perk.org/>.

Acknowledgements. We would like to express our sincere gratitude to Alessandro Budroni, Victor Mateu and Lucas Perin for fruitful discussions and their contribution to the implementation of the scheme.

Conflicts of interest. All authors of this paper are involved in the PERK submission to the NIST post-quantum additional call for digital signatures. The extended list of contributors to the PERK submission is available for consultation at <https://pqc-perk.org/>. There are no other competing interests to declare.

References

1. Adj, G., Rivera-Zamarripa, L., Verbel, J.: MinRank in the head: Short signatures from zero-knowledge proofs. Cryptology ePrint Archive, Report 2022/1501 (2022), <https://eprint.iacr.org/2022/1501>

2. Aguilar-Melchor, C., Gama, N., Howe, J., Hülsing, A., Joseph, D., Yue, D.: The return of the sdith. In: Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part V. pp. 564–596. Springer (2023)
3. Alekhnovich, M.: More on average case vs approximation complexity. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science. p. 298. FOCS '03, IEEE Computer Society, USA (2003)
4. Attema, T., Cramer, R., Kohl, L.: A compressed Σ -protocol theory for lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 549–579. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_19
5. Attema, T., Fehr, S.: Parallel repetition of (k_1, \dots, k_μ) -special-sound multi-round interactive proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 415–443. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_15
6. Attema, T., Fehr, S., Kloof, M.: Fiat-shamir transformation of multi-round interactive proofs. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 113–142. Springer, Heidelberg (Nov 2022). https://doi.org/10.1007/978-3-031-22318-1_5
7. Baldi, M., Bitzer, S., Pavoni, A., Santini, P., Wachter-Zeh, A., Weger, V.: Zero knowledge protocols and signatures from the restricted syndrome decoding problem. Cryptology ePrint Archive (2023)
8. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H.: On the security of the permuted kernel identification scheme. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 305–311. Springer, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_21
9. Bellare, M., Davis, H., Günther, F.: Separate your domains: NIST PQC KEMs, oracle cloning and read-only indistinguishability. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 3–32. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_1
10. Berger, T.P., Gueye, C.T., Klamti, J.B.: A np-complete problem in coding theory with application to code based cryptography. In: Codes, Cryptology and Information Security: Second International Conference, C2SI 2017, Rabat, Morocco, April 10–12, 2017, Proceedings-In Honor of Claude Carlet. pp. 230–237. Springer (2017)
11. Beullens, W.: Not enough LESS: An improved algorithm for solving code equivalence problems over \mathbb{F}_q . In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 387–403. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_15
12. Beullens, W.: Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 183–211. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45727-3_7
13. Beullens, W., Faugère, J.C., Koussa, E., Macario-Rat, G., Patarin, J., Perret, L.: PKP-based signature scheme. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) INDOCRYPT 2019. LNCS, vol. 11898, pp. 3–22. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-35423-7_1
14. Beullens, W., Feo, L.D., Galbraith, S.D., Petit, C.: Proving knowledge of isogenies – a survey. Cryptology ePrint Archive, Paper 2023/671 (2023), <https://eprint.iacr.org/2023/671>, <https://eprint.iacr.org/2023/671>

15. Bidoux, L., Gaborit, P.: Compact post-quantum signatures from proofs of knowledge leveraging structure for the PKP, SD and RSD problems. In: Codes, Cryptology and Information Security (C2SI). pp. 10–42. Springer (2023)
16. Bidoux, L., Gaborit, P., Kulkarni, M., Mateu, V.: Code-based signatures from new proofs of knowledge for the syndrome decoding problem. *Designs, Codes and Cryptography* **91**(2), 497–544 (2023)
17. Chase, M., Derler, D., Goldfeder, S., Kales, D., Katz, J., Kolesnikov, V., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Wang, X., Zaverucha, G.: The picnic signature scheme design document (version 3.0). Available at <https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/Picnic-Round3.zip> (Last checked on: May 24, 2023) (2020)
18. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your MEDS: Digital Signatures from Matrix Code Equivalence. *Cryptology ePrint Archive* (2022)
19. Delfs, C., Galbraith, S.D.: Computing Isogenies between Supersingular Elliptic Curves over \mathbb{F}_p . *Des. Codes Cryptography* **78**(2), 425–440 (feb 2016). <https://doi.org/10.1007/s10623-014-0010-1>, <https://doi.org/10.1007/s10623-014-0010-1>
20. Esser, A., Verbel, J., Zweyding, F., Bellini, E.: *CryptographicEstimators*: a software library for cryptographic hardness estimation. *Cryptology ePrint Archive* (2023)
21. Feneuil, T.: Building MPCitH-based signatures from MQ, MinRank, rank SD and PKP. *Cryptology ePrint Archive*, Report 2022/1512 (2022), <https://eprint.iacr.org/2022/1512>
22. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 541–572. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_19
23. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. *Cryptology ePrint Archive*, Report 2022/188 (2022), <https://eprint.iacr.org/2022/188>
24. Feneuil, T., Joux, A., Rivain, M.: Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Designs, Codes and Cryptography* **91**(2), 563–608 (2023)
25. Feneuil, T., Maire, J., Rivain, M., Vergnaud, D.: Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part II. LNCS, vol. 13792, pp. 371–402. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22966-4_13
26. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO’86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
27. Gaborit, P., Hauteville, A., Phan, D.H., Tillich, J.P.: Identity-based encryption from codes with rank metric. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 194–224. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9_7
28. Gaborit, P., Zémor, G.: On the hardness of the decoding and the minimum distance problems for rank codes. *IEEE Transactions on Information Theory* **62**(12), 7245–7252 (2016). <https://doi.org/10.1109/TIT.2016.2616127>

29. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). W. H. Freeman (1979)
30. Georgiades, J.: Some remarks on the security of the identification scheme based on permuted kernels. *Journal of Cryptology* **5**(2), 133–137 (Jan 1992). <https://doi.org/10.1007/BF00193565>
31. Gueron, S., Persichetti, E., Santini, P.: Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptography* **6**(1), 5 (2022)
32. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbea, B., Beullens, W.: SPHINCS⁺. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
33. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007). <https://doi.org/10.1145/1250790.1250794>
34. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing* **39**(3), 1121–1152 (2009). <https://doi.org/10.1137/080725398>, <https://doi.org/10.1137/080725398>
35. Jaulmes, É., Joux, A.: Cryptanalysis of PKP: A new approach. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 165–172. Springer, Heidelberg (Feb 2001). https://doi.org/10.1007/3-540-44586-2_12
36. Kales, D., Zaverucha, G.: An attack on some signature schemes constructed from five-pass identification schemes. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20. LNCS, vol. 12579, pp. 3–22. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-65411-5_1
37. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mamman, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018). <https://doi.org/10.1145/3243734.3243805>
38. Koussa, E., Macario-Rat, G., Patarin, J.: On the complexity of the permuted kernel problem. *Cryptology ePrint Archive*, Report 2019/412 (2019), <https://eprint.iacr.org/2019/412>
39. Lampe, R., Patarin, J.: Analysis of some natural variants of the pkp algorithm. *Cryptology ePrint Archive* (2011)
40. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
41. NIST: Post-quantum cryptography standardization (2017), available at <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>
42. NIST: Call for additional digital signature schemes for the post-quantum cryptography standardization process (2022), available at <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf>

43. Overbeck, R., Sendrier, N.: Code-based cryptography, pp. 95–145. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_4, https://doi.org/10.1007/978-3-540-88702-7_4
44. Paiva, T.B., Terada, R.: Cryptanalysis of the binary permuted kernel problem. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 21, Part II. LNCS, vol. 12727, pp. 396–423. Springer, Heidelberg (Jun 2021). https://doi.org/10.1007/978-3-030-78375-4_16
45. Patarin, J., Chauvaud, P.: Improved algorithms for the permuted kernel problem. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 391–402. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_33
46. Peikert, C.: A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939 (2015), <https://eprint.iacr.org/2015/939>
47. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **8**(5), 5–9 (1962)
48. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
49. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**(6) (sep 2009). <https://doi.org/10.1145/1568318.1568324>, <https://doi.org/10.1145/1568318.1568324>
50. Santini, P., Baldi, M., Chiaraluce, F.: Computational hardness of the permuted kernel and subcode equivalence problems. Cryptology ePrint Archive, Report 2022/1749 (2022), <https://eprint.iacr.org/2022/1749>
51. Shamir, A.: An efficient identification scheme based on permuted kernels (extended abstract) (rump session). In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 606–609. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_54
52. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). <https://doi.org/10.1109/SFCS.1994.365700>
53. Wang, W.: Shorter signatures from MQ. Cryptology ePrint Archive, Report 2022/344 (2022), <https://eprint.iacr.org/2022/344>

Supplementary Material

A Background and Extra Definitions

In this section we present the definitions of standard cryptographic primitives such as pseudorandom generators (PRG), collision-resistant hash functions (CRHF), commitment schemes, and Merkle trees. Followed by definitions related to generalizing the notion of special soundness. We conclude the section by presenting some details related to the random oracle model, and Fiat-Shamir transformation.

A.1 Standard Cryptographic Primitives

Definition A.1 (Pseudorandom Generator (PRG)). Let p be a polynomial and let G be a deterministic polynomial-time algorithm such that for any $\lambda \in \mathbb{N}$ and any input $s \in \{0, 1\}^\lambda$, the result $G(s)$ is a string of length $p(\lambda)$. We say that G is a pseudorandom generator if the following conditions hold:

1. *Expansion:* For every $\lambda \in \mathbb{N}$ it holds that $p(\lambda) > \lambda$.
2. *Pseudorandomness:* For any PPT algorithm D , there is a negligible function negl such that

$$|\Pr[D(G(s)) = 1] - \Pr[D(r) = 1]| \leq \text{negl}(\lambda)$$

where the first probability is taken over the uniform choice of $s \in \{0, 1\}^\lambda$ and the randomness of D , and the second probability is taken over the choice of $r \in \{0, 1\}^{p(\lambda)}$ and the randomness of D .

We say G is $(t, \epsilon_{\text{PRG}})$ -secure if for every D running in time at most $t(\lambda)$ the success probability of D is upper bounded by some function $\epsilon_{\text{PRG}}(\lambda)$.

Definition A.2 (Collision-Resistant Hash Functions (CRHF)). Let ℓ, κ be polynomials and let $\mathcal{H} = \{H_k : \{0, 1\}^* \rightarrow \{0, 1\}^{\ell(\lambda)}; k \in \{0, 1\}^{\kappa(\lambda)}\}_\lambda$ be a family of functions indexed by $\lambda \in \mathbb{N}$. We say that \mathcal{H} is collision-resistant if there exists a negligible function negl such that, for any PPT algorithm \mathcal{A} it holds that,

$$\Pr \left[\begin{array}{l} x \neq x' \wedge \\ H_k(x) = H_k(x') \end{array} \middle| \begin{array}{l} k \xleftarrow{\$} \{0, 1\}^{\kappa(\lambda)}; \\ (x, x') \leftarrow \mathcal{A}(k) \end{array} \right] \leq \text{negl}(\lambda).$$

Definition A.3 (Commitment Scheme). A commitment scheme is a tuple of algorithms $(\text{Com}, \text{Open})$ such that $\text{Com}(r, m)$ returns a commitment c for the message m and randomness r while $\text{Open}(c, r, m)$ returns either 1 (accept) or 0 (reject). A commitment scheme is said to be correct if:

$$\Pr [b = 1 \mid c \leftarrow \text{Com}(r, m), b \leftarrow \text{Open}(c, r, m)] = 1.$$

Definition A.4 (Computationally Hiding). Let (m_0, m_1) be a pair of messages, the advantage of \mathcal{A} against the hiding experiment is defined as:

$$\text{Adv}_{\mathcal{A}}^{\text{hiding}}(1^\lambda) = \left| \Pr \left[b = b' \mid \begin{array}{l} b \xleftarrow{\$} \{0, 1\}, r \xleftarrow{\$} \{0, 1\}^\lambda \\ c \leftarrow \text{Com}(r, m_b), b' \leftarrow \mathcal{A}.\text{guess}(c) \end{array} \right] - \frac{1}{2} \right|.$$

A commitment scheme is computationally hiding if for all PPT adversaries \mathcal{A} and every pair of messages (m_0, m_1) , $\text{Adv}_{\mathcal{A}}^{\text{hiding}}(1^\lambda)$ is negligible in λ .

We say Com is $(t, \epsilon_{\text{Com}})$ -secure if for every \mathcal{A} running in time at most $t(\lambda)$ the success probability of \mathcal{A} is upper bounded by some function $\epsilon_{\text{Com}}(\lambda)$.

Definition A.5 (Computationally Binding). *The advantage of an adversary \mathcal{A} against the commitment binding experiment is defined as:*

$$\text{Adv}_{\mathcal{A}}^{\text{binding}}(1^\lambda) = \Pr \left[\begin{array}{l} m_0 \neq m_1 \\ 1 \leftarrow \text{Open}(c, r_0, m_0) \\ 1 \leftarrow \text{Open}(c, r_1, m_1) \end{array} \middle| (c, r_0, r_1, m_0, m_1) \leftarrow \mathcal{A}.\text{choose}(1^\lambda) \right].$$

A commitment scheme is computationally binding if for all PPT adversaries \mathcal{A} , $\text{Adv}_{\mathcal{A}}^{\text{binding}}(1^\lambda)$ is negligible in λ .

A.1.1 Merkle Trees Merkle trees can be used in our context to compress randomness seeds as suggested in [37]. Suppose a party needs to generate N seeds and then to send only $N - 1$ of those seeds (without knowing in advance which seed should not be sent). The principle is to build a binary tree of depth $\lceil \log_2(N) \rceil$. The root of the tree is labeled with a master seed θ . The rest of the tree is labeled inductively by using a PRG of double extension on each parent node and splitting the output on the left and right children.

To reveal all seeds except seed number $i \in [N]$, the principle is to reveal the labels on the siblings of the paths from the root of the tree to leave i . It allows to reconstruct all seeds but seed number i at the cost of communicating $\lceil \log_2(N) \rceil$ labels, which is more effective than communicating $N - 1$ seeds.

A.2 Interactive Proofs and Special Soundness

An interactive proof for relation R aims for a prover P to convince a verifier V that a statement x admits a witness, or even that the prover *knows* a witness $w \in R(x)$.

Definition A.6 (Interactive proof (cf. [5])). *An interactive proof $\Pi = (\mathsf{P}, \mathsf{V})$ for relation R is an interactive protocol between two probabilistic machines, a prover P and a polynomial time verifier V . Both P and V take as public input a statement x and, additionally, P takes as private input a witness $w \in R(x)$, which is denoted as $(\mathsf{P}(w), \mathsf{V})(x)$. The verifier V either accepts or rejects the prover's claim of knowing a witness for x , this decision by the verifier is considered the output of the protocol. The set of all messages exchanged in the protocol execution is called a transcript and is denoted $\langle \mathsf{P}(x, w), \mathsf{V}(x) \rangle$. We call the either accepting (or resp. rejecting) based on whether the verifier accepts (or rejects) the prover's claim.*

We assume that the prover sends the first and the last message in any interactive proof. Hence, the number of messages is always an odd number $2\mu + 1$. We also say Π is a $(2\mu + 1)$ -round proof. It is represented in the following figure.

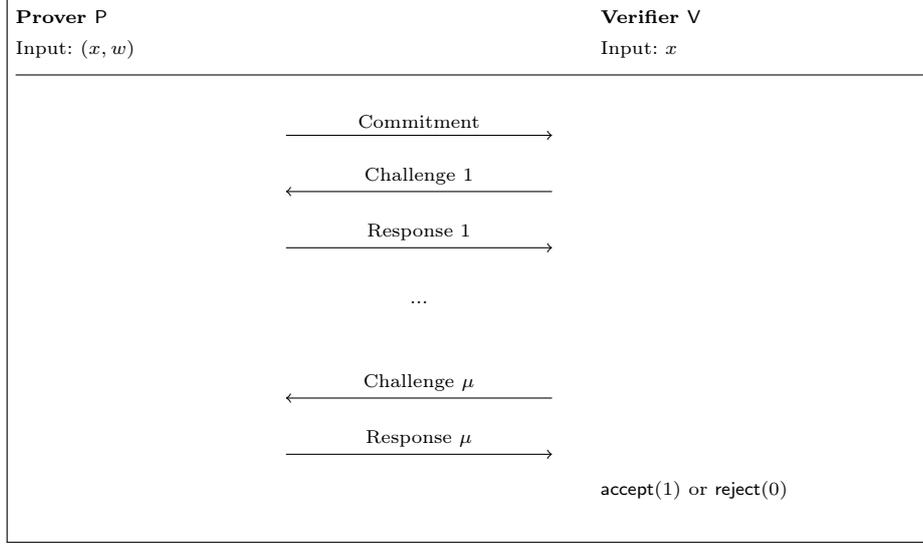


Fig. 5: $(2\mu + 1)$ -round Σ interactive proof

An interactive proof Π is *complete* if the verifier V accepts honest executions with a public-private input pair $(x; w) \in R$ with high probability. It is *sound* if the verifier rejects the false statements $x \notin L_R$ with high probability. In this work, we follow the presentation of [5] and do not require these properties as part of definition of interactive proofs, but consider them as desirable additional security properties.

Definition A.7 (Completeness (cf. [5])). An interactive proof $\Pi = (P, V)$ for relation R is complete with completeness error $\rho : \{0, 1\}^* \rightarrow [0, 1]$ if for every $(x; w) \in R$,

$$\Pr[(P(w), V)(x) = \text{reject}] \leq \rho(x).$$

If $\rho(x) = 0$ for all $x \in L_R$, then Π is said to be perfectly complete.

Definition A.8 (Soundness (cf. [5])). An interactive protocol $\Pi = (P, V)$ for relation R is sound with soundness error $\sigma : \{0, 1\}^* \rightarrow [0, 1]$ if for every $x \notin L_R$ and every prover P^* ,

$$\Pr[(P^*, V)(x) = \text{accept}] \leq \sigma(x).$$

An interactive proof which is complete and sound allows a prover to convince a verifier that the statement x is true, i.e., $x \in L_R$. However, this does not necessarily convince a verifier that the prover actually “knows” the witness $w \in R(x)$. This stronger property is captured by the notion of *knowledge soundness*. Informally, knowledge soundness guarantees that if a prover convinces a verifier

about the validity of some statement x with sufficiently high probability, then the prover can actually compute a witness $w \in R(x)$ with high probability.⁵

Next, we present the definitions related to the special-soundness in the generalized scenario. In order to generalize k -special-soundness to multi-round protocols we will introduce the notion of a tree of transcripts following the definitions given in [4].

Definition A.9 (Tree of Transcripts (cf. [5])). Let $k_1, \dots, k_\mu \in \mathbb{N}$. A (k_1, \dots, k_μ) -tree of transcripts for a $(2\mu + 1)$ -round public-coin protocol $\Pi = (\mathcal{P}, \mathcal{V})$ is a set of $K = \prod_{i=1}^\mu k_i$ transcripts arranged in the following tree structure. The nodes in this tree correspond to the prover's messages and the edges to the verifier's challenges. Every node at depth i has precisely k_i children corresponding to k_i pairwise distinct challenges. Every transcript corresponds to exactly one path from the root to a leaf node. For a graphical representation we refer to Fig. 6. We refer to the corresponding tree of challenges as a (k_1, \dots, k_μ) -tree of challenges.

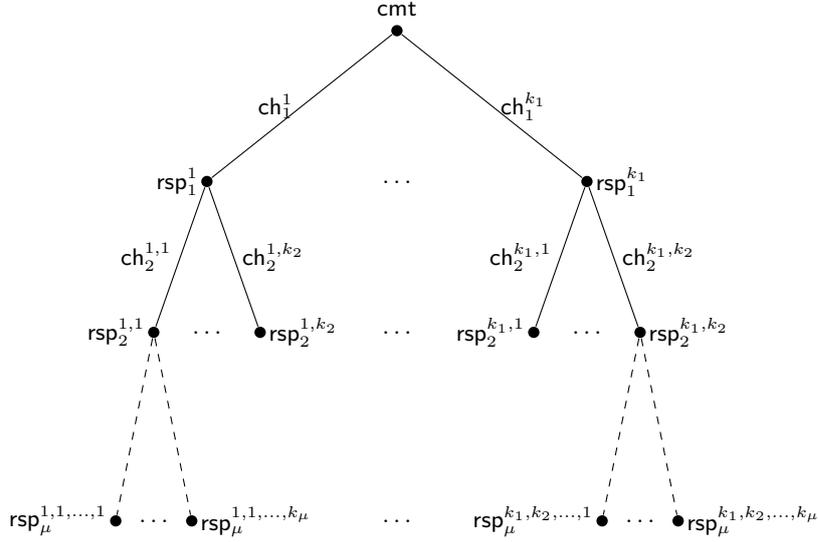


Fig. 6: (k_1, k_2, \dots, k_μ) tree of transcripts of a $(2\mu + 1)$ -round public-coin protocol

We will also write $\mathbf{k} = (k_1, \dots, k_\mu) \in \mathbb{N}^\mu$ and refer to a \mathbf{k} -tree of transcripts.

Definition A.10 ((k_1, \dots, k_μ) -out-of- (N_1, \dots, N_μ) Special Soundness (cf. [5])). Let $k_1, \dots, k_\mu, N_1, \dots, N_\mu \in \mathbb{N}$. A $(2\mu + 1)$ -round public-coin protocol

⁵ Since the protocol presented in this work only achieves computational soundness, and is secure when the prover runs in *polynomial time*, technically our protocol is an *argument of knowledge*. However, we avoid this distinction for simplicity.

$\Pi = (\mathsf{P}, \mathsf{V})$ for a relation R , where V samples the i -th challenge from a set of cardinality $N_i \geq k_i$ for $1 \leq i \leq \mu$, is (k_1, \dots, k_μ) -out-of- (N_1, \dots, N_μ) special-sound if there exists a polynomial time algorithm that, on a input statement x and a (k_1, \dots, k_μ) -tree of accepting transcripts outputs a witness $w \in R(x)$. We also say Π is (k_1, \dots, k_μ) special-sound.

A.3 Fiat-Shamir Transformation

In this section, we explain the random oracle model and Fiat-Shamir transformation used for transforming interactive protocols into non-interactive ones. We closely follow the presentation of [6, Section 2.3] in the following exposition.

In the *random oracle model (ROM)*, algorithms have black-box (or input-output) access to an oracle $\text{RO} : \{0, 1\}^* \rightarrow \mathcal{Z}$, called as *random oracle*, which is instantiated with a uniform random function with domain $\{0, 1\}^*$ and codomain \mathcal{Z} . Generally, $\mathcal{Z} = \{0, 1\}^\eta$ for some $\eta \in \mathbb{N}$ related to the security parameter. In practice, RO can be implemented by lazy sampling, which means for each input string $x \in \{0, 1\}^*$, $\text{RO}(x)$ is sampled uniform randomly from \mathcal{Z} and then fixed. To avoid technical difficulties, we limit the domain from $\{0, 1\}^*$ to $\{0, 1\}^{\leq \ell}$, the finite set of all bitstrings of length at most ℓ , for a sufficiently large $\ell \in \mathbb{N}$.

An algorithm \mathcal{A}^{RO} that is given black-box access to a random oracle is called a *random oracle algorithm*. We say \mathcal{A} is a Q -query random-oracle algorithm, if it makes at most Q queries to RO (independent of RO).

A natural extension of the ROM is when \mathcal{A} is given access to *multiple independent* random oracles $\text{RO}_1, \text{RO}_2, \dots, \text{RO}_\mu$, possibly with different codomains. In practice, these random oracles can be instantiated by a single random oracle $\text{RO} : \{0, 1\}^* \rightarrow \{0, 1\}^\eta$ using the standard techniques for domain separation (refer to [9] for more details) and for sampling random elements from non-binary sets.

The Fiat-Shamir transform [26], turns a public-coin interactive proof into a non-interactive proof in random oracle model. The general idea of this transformation is to compute the i -th challenge message ch_i as a hash of the i -th prover message a_i along with (partial) communication transcript generated till that point. For a Σ -protocol, the challenge ch is computed as $\text{ch} := \text{H}(\text{cmt})$ or as $\text{ch} := \text{H}(x, \text{cmt})$, where the former is sufficient for *static* security, where the statement x is given as input to the dishonest prover, and the latter is necessary for *adaptive* security, where the dishonest prover can choose the statement x for which it wants to forge a proof.

For multi-round public-coin interactive proofs, there is some degree of freedom in the computation of the i -th challenge. For concreteness we consider a particular version where all previous messages are hashed along with the current message.

Let $\Pi = (\mathsf{P}, \mathsf{V})$ be a $(2\mu + 1)$ -round public-coin interactive proof, where the challenge for the i -th round is sampled from set \mathcal{CH}_i . For simplicity, we consider μ random oracles $\text{RO}_i : \{0, 1\}^{\leq \ell} \rightarrow \mathcal{CH}_i$ that map into the respective challenge spaces.

Definition A.11 (Fiat-Shamir Transformation (cf. [6])). *The static Fiat-Shamir transformation $\text{FS}[II] = (\text{P}_{\text{fs}}, \text{V}_{\text{fs}})$ is non-interactive proof in ROM, where $\text{P}_{\text{fs}}^{\text{RO}_1, \text{RO}_2, \dots, \text{RO}_\mu}(x; w)$ runs $\text{P}(x; w)$ but instead of asking the verifier for the challenge ch_i on message a_i , the challenges are computed as*

$$\text{ch}_i = \text{RO}_i(a_1, a_2, \dots, a_{i-1}, a_i); \quad (5)$$

the output is then the proof $\pi = (a_1, \dots, a_{\mu+1})$. On input a statement x and a proof $\pi = (a_1, \dots, a_{\mu+1})$, $\text{P}_{\text{fs}}^{\text{RO}_1, \text{RO}_2, \dots, \text{RO}_\mu}(x, \pi)$ accepts if, for ch_i as above V accepts the transcript $(a_1, \text{ch}_1, \dots, a_\mu, \text{ch}_\mu, a_{\mu+1})$ on input x .

If the challenges are computed as

$$\text{ch}_i = \text{RO}_i(x, a_1, \text{ch}_1, \dots, a_{i-1}, \text{ch}_{i-1}, a_i); \quad (6)$$

the resulting non-interactive proof in ROM is called as the adaptive Fiat-Shamir transformation.

B Security Proofs for PoK

B.1 Proof of Theorem 3.2

We restate the Theorem 3.2 below and follow it by its proof.

Theorem 3.2 (Knowledge Soundness). *The protocol presented in Fig. 1 is knowledge sound with knowledge error*

$$\varepsilon_{\text{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}.$$

Proof (Proof of Theorem 3.2). Before proving the knowledge soundness of our protocol, we will first prove the following useful lemma.

Lemma B.1 ((2,2)-special soundness). *The protocol shown in Fig. 1 is (2,2)-special sound.*

Proof (Proof of Lemma B.1). **(2,2)-special soundness.**

Following Definition A.10 the protocol is called (2,2)-special sound if there exists an efficient knowledge extractor Ext which on an input statement $(\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$ and a (2,2)-accepting tree of transcripts (See Definition A.9) returns a solution of the r-IPKP instance defined by $(\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$. We now show such an extractor which takes 4 accepting transcripts associated with challenges $(\kappa, \alpha_1), (\kappa, \alpha_2), (\kappa', \alpha'_1), (\kappa', \alpha'_2)$ such that $\kappa = (\kappa_i)_{i \in [1, t]}$, $\kappa' = (\kappa'_i)_{i \in [1, t]}$ as well as $\kappa \neq \kappa'$, $\alpha_1 \neq \alpha_2$, and $\alpha'_1 \neq \alpha'_2$, and outputs a solution to the r-IPKP instance defined by $(\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$.

Let $z_2^{(\kappa^*, \alpha^*)}$ denote the response z_2 computed as shown in Fig. 1 when the first and second challenges are κ^* and α^* respectively. Note that, $z_2^{(\kappa, \alpha_1)}$ contains

all the seeds θ_i for $i \in [1, N]$ except $i = \alpha_1$. Therefore, the extractor has access to all the seeds θ_i for $i \in [1, N]$ since it knows both $z_2^{(\kappa, \alpha_1)}$ as well as $z_2^{(\kappa, \alpha_2)}$ and $\alpha_1 \neq \alpha_2$. It can compute $(\bar{\pi}_i^{(\kappa)}, \bar{\mathbf{v}}_i^{(\kappa)})_{i \in [1, N]}$ and $(\bar{\pi}_i^{(\kappa')}, \bar{\mathbf{v}}_i^{(\kappa')})_{i \in [1, N]}$ from $(z_2^{(\kappa, \alpha_1)})_{i \in [1, 2]}$ and $(z_2^{(\kappa', \alpha'_i)})_{i \in [1, 2]}$ respectively.

Also, note that the first message $h_1 = \mathbf{H}(\text{cmt}_1, (\text{cmt}_{1,i})_{i \in [1, N]})$ is common to all the 4 transcripts. Since we assume that \mathbf{H} is a collision-resistant hash function, it means that the initial commitments $(\text{cmt}_1, (\text{cmt}_{1,i})_{i \in [1, N]})$ are all same in the 4 transcripts. From the binding property of the commitments $(\text{cmt}_{1,i})_{i \in [1, N]}$, we know that

$$(\bar{\pi}_i, \bar{\mathbf{v}}_i)_{i \in [1, N]} = (\bar{\pi}_i^{(\kappa)}, \bar{\mathbf{v}}_i^{(\kappa)})_{i \in [1, N]} = (\bar{\pi}_i^{(\kappa')}, \bar{\mathbf{v}}_i^{(\kappa')})_{i \in [1, N]}.$$

The knowledge extractor Ext computes the solution as

1. Compute $(\bar{\pi}_i)_{i \in [1, n]}$ from $z_2^{(\kappa, \alpha_1)}$ and $z_2^{(\kappa, \alpha_2)}$
2. Output $\bar{\pi} = \bar{\pi}_N \circ \dots \circ \bar{\pi}_1$

Let us now check the validity of this solution output by the extractor. By construction, we know that $\bar{\mathbf{s}}_0^{(\kappa, \alpha_1)} = \bar{\mathbf{s}}_0^{(\kappa, \alpha_2)} = \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i$. Also, for all $i \in [1, N] \setminus \alpha_1$, $\bar{\mathbf{s}}_i^{(\kappa, \alpha_1)} = \bar{\pi}_i \left[\bar{\mathbf{s}}_{i-1}^{(\kappa, \alpha_1)} \right] + \bar{\mathbf{v}}_i$. And for all $i \in [1, N] \setminus \alpha_2$, $\bar{\mathbf{s}}_i^{(\kappa, \alpha_2)} = \bar{\pi}_i \left[\bar{\mathbf{s}}_{i-1}^{(\kappa, \alpha_2)} \right] + \bar{\mathbf{v}}_i$. Since the transcripts are accepting and \mathbf{V} checks h_2 computed as $h_2 = \mathbf{H}((\mathbf{s}_i)_{i \in [1, N]})$, due to the collision-resistance property of \mathbf{H} , it follows that for all $i \in [1, N]$, $\bar{\mathbf{s}}_i^{(\kappa)} = \bar{\pi}_i \left[\bar{\mathbf{s}}_{i-1}^{(\kappa)} \right] + \bar{\mathbf{v}}_i$, this implies $\bar{\mathbf{s}}_N^{(\kappa)} = \bar{\pi} \left[\sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i \right] + \bar{\mathbf{v}}$.

Following a similar argument, we know that $\bar{\mathbf{s}}_N^{(\kappa')} = \bar{\pi} \left[\sum_{i \in [1, t]} \kappa'_i \cdot \mathbf{x}_i \right] + \bar{\mathbf{v}}$. Based on the binding property of commitment cmt_1 and using the fact that the transcripts are accepting, we can write

$$\begin{aligned} \mathbf{H} \bar{\mathbf{s}}_N^{(\kappa)} - \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{y}_i &= \mathbf{H} \bar{\mathbf{s}}_N^{(\kappa')} - \sum_{i \in [1, t]} \kappa'_i \cdot \mathbf{y}_i \\ \implies \mathbf{H} \left(\bar{\pi} \left[\sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i \right] + \bar{\mathbf{v}} \right) - \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{y}_i &= \mathbf{H} \left(\bar{\pi} \left[\sum_{i \in [1, t]} \kappa'_i \cdot \mathbf{x}_i \right] + \bar{\mathbf{v}} \right) - \sum_{i \in [1, t]} \kappa'_i \cdot \mathbf{y}_i \\ \implies \mathbf{H} \left(\bar{\pi} \left[\sum_{i \in [1, t]} (\kappa_i - \kappa'_i) \cdot \mathbf{x}_i \right] \right) &= \sum_{i \in [1, t]} (\kappa_i - \kappa'_i) \cdot \mathbf{y}_i \end{aligned}$$

This implies that $\bar{\pi}$ is a solution of the considered r-IPKP problem. \square

We can now apply the result of Theorem 2.1 to Lemma B.1, which concludes the proof. \square

B.2 Proof of Theorem 3.3

The following proof is inspired from the proof of [17, Lemma 6.1] and [22, 23, Theorem 3]. We now show that the protocol described in Fig. 1, Section 3, satisfies the special honest-verifier zero knowledge property. We assume that the commitment algorithm $\text{Com}(\cdot)$ outputs $\ell(\lambda)$ -bit strings as output for some polynomial ℓ . We restate the Theorem 3.3 below and follow it by its proof.

Theorem 3.3 (Special Honest-Verifier Zero Knowledge). *Assume that there exists a $(\mathfrak{t}, \epsilon_{\text{PRG}})$ -secure PRG, and the commitment scheme Com is $(\mathfrak{t}, \epsilon_{\text{Com}})$ -hiding. Then there exists an efficient simulator Sim which, outputs a transcript such that no distinguisher running in time at most $\mathfrak{t}(\lambda)$ can distinguish between the transcript produced by Sim and a real transcript obtained by honest execution of the protocol in Fig. 1 with probability better than $(\epsilon_{\text{PRG}}(\lambda) + \epsilon_{\text{Com}}(\lambda))$.*

Proof (Proof of Theorem 3.3). We begin by describing an efficient simulator Sim which outputs a transcript which is indistinguishable from a real transcript obtained by honest execution of the protocol. Sim on input $x = (\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$ works as follows:

Note that the simulator Sim runs in polynomial-time and the challenges sampled in Step 1 are distributed identically to the real world execution since the verifier also samples the challenges uniformly at random. We now show that the transcript output by Sim and a real transcript obtained by honest execution of the protocol in Fig. 1 with challenges (κ, α^*) cannot be distinguished with probability better than $(\epsilon_{\text{PRG}}(\lambda) + \epsilon_{\text{Com}}(\lambda))$ by any distinguisher running in time at most $\mathfrak{t}(\lambda)$. We consider the following sequence of simulators:

Simulator 0 (real world). This simulator takes the statement $x = (\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$, witness π , and the challenges (κ, α^*) as input. It then runs the protocol in Fig. 1 honestly and outputs the transcript. This transcript is identically distributed as a real-world transcript.

Simulator 1. Simulator 1 works exactly same as Simulator 0 except that instead of computing cmt_{1, α^*} as in the real protocol, it samples a uniform string as $\text{cmt}_{1, \alpha^*} \xleftarrow{\$} \{0, 1\}^{\ell(\lambda)}$. The probability of distinguishing Simulator 0 from Simulator 1 by any distinguisher running in time at most $\mathfrak{t}(\lambda)$ is upper bounded by $\epsilon_{\text{Com}}(\lambda)$.

Simulator 2. The only difference between Simulator 1 and Simulator 2 is that, Simulator 2 samples $\pi_{\alpha^*} \xleftarrow{\$} \mathcal{S}_n$ and $\mathbf{v}_{\alpha^*} \xleftarrow{\$} \mathbb{F}_q^n$ uniformly at random instead of using the seed-derived randomness from the seed θ . The probability of distinguishing Simulator 2 from Simulator 1 by any distinguisher running in time at most $\mathfrak{t}(\lambda)$ is upper bounded by $\epsilon_{\text{PRG}}(\lambda)$.

Simulator 3 (Sim). Simulator 3 takes the statement $x = (\mathbf{H}, (\mathbf{x}_i)_{i \in [1, t]}, (\mathbf{y}_i)_{i \in [1, t]})$, and works as Sim defined in Fig. 7. Note that, this simulator does not depend on

1. Sample $\kappa \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$ and $\alpha^* \xleftarrow{\$} [N]$
2. Sample $\theta \xleftarrow{\$} \{0, 1\}^\lambda$
3. For each $i \in [1, N] \setminus \{\alpha^*\}$
 - ◊ $\theta_i \xleftarrow{\$, \theta} \{0, 1\}^\lambda$, $\phi_i \xleftarrow{\$, \theta_i} \{0, 1\}^\lambda$, $r_{1,i} \xleftarrow{\$, \theta_i} \{0, 1\}^\lambda$
 - ◊ **if** $i \neq 1$
 - ▷ $\pi_i \xleftarrow{\$, \phi_i} \mathcal{S}_n$, $\mathbf{v}_i \xleftarrow{\$, \phi_i} \mathbb{F}_q^n$, $\text{cmt}_{1,i} = \text{Com}(r_{1,i}, \phi_i)$
 - ◊ **if** $i = 1$
 - ▷ $\pi_1 \xleftarrow{\$} \mathcal{S}_n$, $\mathbf{v}_1 \xleftarrow{\$, \phi_1} \mathbb{F}_q^n$, $\text{cmt}_{1,1} = \text{Com}(r_{1,1}, \pi_1 \parallel \phi_1)$
4. For $i = \alpha^*$
 - ◊ Sample $\pi_{\alpha^*} \xleftarrow{\$} \mathcal{S}_n$, $\mathbf{v}_{\alpha^*} \xleftarrow{\$} \mathbb{F}_q^n$, $\text{cmt}_{1,\alpha^*} \xleftarrow{\$} \{0, 1\}^{\ell(\lambda)}$
5. Compute r_1 , \mathbf{v} , cmt_1 , h_1 as in the real protocol using the values computed above.
6. Compute $\tilde{\pi} = \pi_N \circ \dots \circ \pi_1$
7. Compute $\tilde{\mathbf{x}}$ such that $\mathbf{H}\tilde{\mathbf{x}} = \sum_{i \in [1,t]} \kappa_i \cdot \mathbf{y}_i$
8. Compute $\mathbf{s}_0 = \sum_{i \in [1,t]} \kappa_i \cdot \mathbf{x}_i$
9. For each $i \in [1, \alpha^* - 1]$
 - ◊ Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$
10. Compute $\mathbf{s}_{\alpha^*} = \pi_{\alpha^*}[\mathbf{s}_{\alpha^*-1}] + \mathbf{v}_{\alpha^*} + \pi_{\alpha^*+1}^{-1} \circ \dots \circ \pi_N^{-1} \left[\tilde{\mathbf{x}} - \tilde{\pi} \left[\sum_{i \in [1,t]} \kappa_i \cdot \mathbf{x}_i \right] \right]$
11. For each $i \in [\alpha^* + 1, N]$
 - ◊ Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$
12. Compute h_2 , z_1 , z_2 , rsp as in the real protocol using the values computed above.
13. Output $(x, h_1, (\kappa_i)_{i \in [1,t]}, h_2, \alpha^*, \text{rsp})$

Fig. 7: Simulator Sim for generating indistinguishable transcripts without knowledge of secret witness π

the witness π . Also, Sim first samples the challenges (κ, α^*) uniform randomly (this is identical to honest verifier in real world).

If $\alpha^* = 1$, then Simulator 2 and Sim work exactly same till Step 5 of Sim . Therefore, h_1 is distributed identically in both the transcripts. Also, \mathbf{s}_0 is computed honestly by Sim and hence matches with that computed by Simulator 2. While computing \mathbf{s}_1 , both Simulator 2 and Sim add \mathbf{v}_1 to it. However, \mathbf{v}_1 is sampled uniformly at random by both Simulator 2 and Sim . Hence, \mathbf{s}_1 is also distributed identically in both the transcripts. Step 11 of Sim works exactly as Simulator 2 which means h_2 is also distributed identically in both the transcripts. The response rsp in this case is $\text{rsp} = (\mathbf{s}_1, (r_1 \parallel \theta_i)_{i \in [2,N]}, \text{cmt}_{1,1})$. As explained above \mathbf{s}_1 is uniform random and distributed identically in both transcripts. The

seeds $(\theta_i)_{i \in [2, N]}$ and randomness r_1 are computed identically by both the simulators since they work exactly the same way till Step 5 of Sim. Also, $\text{cmt}_{1,1}$ is sampled uniformly at random in both experiments (refer Simulator 1). Therefore, the transcript $(x, h_1, (\kappa_i)_{i \in [1, t]}, h_2, \alpha^*, \text{rsp})$ is distributed identically in Simulator 2 and Sim when $\alpha^* = 1$.

If $\alpha^* \neq 1$, then Simulator 2 and Sim work exactly same till Step 5 of Sim, except for sampling of π_1 . Simulator 2 computes π_1 from witness π , whereas Sim samples π_1 uniformly at random. However, the values $r_1, \mathbf{v}, \text{cmt}_1$, and h_1 are computed independently of π_1 and those are distributed identically in both the transcripts. Also note Simulator 2 computes π_1 by composing it with $\pi_{\alpha^*}^{-1}$. Since π_{α^*} is sampled uniformly at random by Simulator 2, this implies that π_1 computed by Simulator 2 is also uniform random permutation and hence π_1 is also distributed identically. Also, for $i \in [0, \alpha^* - 1]$, the values \mathbf{s}_i are computed honestly by Sim and since π_1 is distributed identically the values \mathbf{s}_i for $i \in [0, \alpha^* - 1]$ are also distributed identically. As in the previous case, while computing \mathbf{s}_{α^*} , both Simulator 2 and Sim add \mathbf{v}_{α^*} to it. However, \mathbf{v}_{α^*} is sampled uniformly at random by both Simulator 2 and Sim. Hence, \mathbf{s}_{α^*} is also distributed identically in both the transcripts. Step 11 of Sim works exactly as Simulator 2 which means h_2 is also distributed identically in both the transcripts. The response rsp in this case is $\text{rsp} = (\mathbf{s}_{\alpha^*}, (\pi_1 || r_1 || \theta_i)_{i \in [1, N] \setminus \alpha^*}, \text{cmt}_{1, \alpha^*})$. As explained above \mathbf{s}_{α^*} is uniform random and distributed identically in both transcripts. The seeds $(\theta_i)_{i \in [1, N] \setminus \alpha^*}$ and randomness r_1 are computed identically by both the simulators since they work exactly the same way till Step 5 of Sim. Also, cmt_{1, α^*} is sampled uniformly at random in both experiments (refer Simulator 1). Therefore, the transcript $(x, h_1, (\kappa_i)_{i \in [1, t]}, h_2, \alpha^*, \text{rsp})$ is distributed identically in Simulator 2 and Sim when $\alpha^* \neq 1$.

Therefore, any distinguisher running in time at most $\mathbf{t}(\lambda)$ cannot distinguish between the real-world transcript and the transcript produced by Sim with probability better than $(\epsilon_{\text{PRG}}(\lambda) + \epsilon_{\text{Com}}(\lambda))$.

□

C Security Proof - PERK

C.1 Proof of Theorem 3.4

We restate the Theorem 3.4 below and follow it by its proof.

Theorem 3.4. *Suppose PRG is $(\mathbf{t}, \epsilon_{\text{PRG}})$ -secure and any adversary running in time $\mathbf{t}(\lambda)$ can solve the the underlying r-IPKP instance with probability at most $\epsilon_{\text{r-IPKP}}$. Model H_0, H_1 , and H_2 as random oracles where H_0, H_1 , and H_2 have 2λ -bit output length. Then chosen-message attacker against the signature scheme (PERK) presented in Fig. 3, running in time $\mathbf{t}(\lambda)$, making q_s signing queries, and making q_0, q_1, q_2 queries, respectively, to the random oracles, succeeds in*

outputting a valid forgery with probability

$$\Pr[\text{Forge}] \leq \frac{(q_0 + \tau \cdot (N + 1) \cdot q_s)^2}{2 \cdot 2^{2\lambda}} + \frac{q_s \cdot (q_0 + q_1 + q_2 + q_s)}{2^{2\lambda}} + \tau \cdot q_s \cdot \epsilon_{\text{PRG}}(\lambda) + \epsilon_{r\text{-IPKP}} + q_2 \cdot \epsilon_{\text{KS}}^{\tau}, \quad (3)$$

where $\epsilon_{\text{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}$.

The following proof is greatly inspired from the proof of the Picnic signature scheme [17, Theorem 6.2] and [22, 23, Theorem 5].

Proof (Proof of Theorem 3.4). Let \mathcal{A} be a EUF-CMA attacker against the signature scheme, which makes q_s queries to the signing oracle. Also, let q_0 , q_1 , and q_2 respectively denote the number of queries made by \mathcal{A} to the random oracles H_0 , H_1 , and H_2 . To prove security we define a sequence of experiments involving \mathcal{A} , starting with an experiment in which \mathcal{A} interacts with the real signature scheme. We let $\Pr_i[\cdot]$ refer to the probability of an event in experiment i . We let $t(\lambda)$ denote the running time of the entire experiment, *i.e.*, including both \mathcal{A} 's running time and the time required to answer signing queries and to verify \mathcal{A} 's output.

Experiment 1. This corresponds to the interaction of \mathcal{A} with the real signature scheme. In more detail: first KeyGen is run to obtain, the secret key $\text{sk} = \pi$ along with the public key $\text{pk} = (\mathbf{H}, (\mathbf{x}_j, \mathbf{y}_j)_{j \in [1, t]})$, and \mathcal{A} is given pk . In addition, we assume that the random oracles H_0 , H_1 , and H_2 are chosen uniformly from the appropriate spaces. \mathcal{A} may make signing queries, which will be answered as in the signature algorithm; \mathcal{A} may also query any of the random oracles. Finally, \mathcal{A} outputs a message-signature pair; we let **Forge** denote the event that the message was not previously queried by \mathcal{A} to its signing oracle, and the signature is valid. Our goal is to upper-bound $\Pr_1[\text{Forge}]$.

Experiment 2. We abort the experiment if, during the course of the experiment, a collision occurs in H_0 . The number of queries to any oracle throughout the experiment (by either the adversary or the signing algorithm) is at most $(q_0 + \tau \cdot (N + 1) \cdot q_s)$. Therefore,

$$|\Pr_1[\text{Forge}] - \Pr_2[\text{Forge}]| \leq \frac{(q_0 + \tau \cdot (N + 1) \cdot q_s)^2}{2 \cdot 2^{2\lambda}}.$$

Experiment 3. We abort the experiment if during the course of the experiment, while answering to a signature query, the sampled salt collides with the value salt in any previous query to H_0 , H_1 , or H_2 . For each single signature query, the probability to abort is upper bounded by $(q_0 + q_1 + q_2 + q_s) / 2^{2\lambda}$. Thus,

$$|\Pr_2[\text{Forge}] - \Pr_3[\text{Forge}]| \leq \frac{q_s \cdot (q_0 + q_1 + q_2 + q_s)}{2^{2\lambda}}.$$

Experiment 4. The difference with the previous experiment is that, when signing a message m we begin by choosing h_1 and h_2 uniformly at random and then we expand them as $(\kappa_j^{(e)})_{e \in [1, \tau], j \in [1, t]}$ and $(\alpha^{(e)})_{e \in [1, \tau]}$. Steps 1, 3, and 5 are computed as before, but in Steps 2 and 4 we simply set the output of H_1 to h_1 and the output of H_2 to h_2 .

The outcome of this experiment compared to the previous one only changes if, in the course of answering a signing query, the query to H_1 or the query to H_2 was ever made before (by either the adversary or as a part of answering some other signing query). But this cannot happen since in such a case Experiment 3 would abort. Thus,

$$\Pr_3[\text{Forge}] = \Pr_4[\text{Forge}].$$

Experiment 5. The difference with the previous experiment is that, for each $e \in [1, \tau]$, we sample $\text{cmt}_{1, \alpha^{(e)}}^{(e)}$ uniformly at random instead of making a query to H_0 .

The only difference between this experiment and the previous experiment occurs if, during the course of answering a signing query, the seed $\theta_{\alpha^{(e)}}^{(e)}$ (for some $e \in [1, \tau]$) was previously queried to H_0 . However, such collisions cannot occur within the same signing query (since indices e and i are part of the input of H_0) and if it occurs from a previous query (signing query or query to H_0) then the experiment aborts (according to the difference introduced in Experiment 3). Thus,

$$\Pr_4[\text{Forge}] = \Pr_5[\text{Forge}].$$

Experiment 6. We again modify the experiment. Now, for $e \in [1, \tau]$ the signer uses the SHVZK simulator Sim (see proof of Theorem 3.3) to generate the views of the parties during the execution of Step 1 and Step 3. We denote by $\text{Sim}_{\text{salt}}(\cdot)$ a call to this simulator which appends salt to the sampled seed θ as input to PRG. This simulation results in $\left\{ \left(\theta_i^{(e)}, \pi_i^{(e)} \right) \right\}_{i \neq \alpha^{(e)}}$ and $(\mathbf{s}_j^{(e)})_{j \in [1, N]}$. Thus the signing queries are now answered as shown in 8.

Note that the secret π is no longer used for generating signatures. Recall that an adversary against Sim has distinguishing advantage $\epsilon_{\text{PRG}}(\lambda)$ (corresponding to execution time $t(\lambda)$), since the commitments are built outside of the simulator. Therefore,

$$|\Pr_5[\text{Forge}] - \Pr_6[\text{Forge}]| \leq \tau \cdot q_s \cdot \epsilon_{\text{PRG}}(\lambda).$$

Experiment 7. At any point during the experiment, we say that the execution e^* of a query

$$h_2 = H_2(\text{salt}, m, \text{pk}, h_1, (\mathbf{s}_i^{(e)})_{e \in [1, \tau], i \in [1, N]})$$

defines a *correct witness* if the following four conditions are fulfilled:

1. h_1 was output by a previous query

$$h_1 = H_1(\text{salt}, m, \text{pk}, (\text{cmt}_1^{(e)}, \text{cmt}_{1, i}^{(e)})_{e \in [1, \tau], i \in [1, N]}),$$

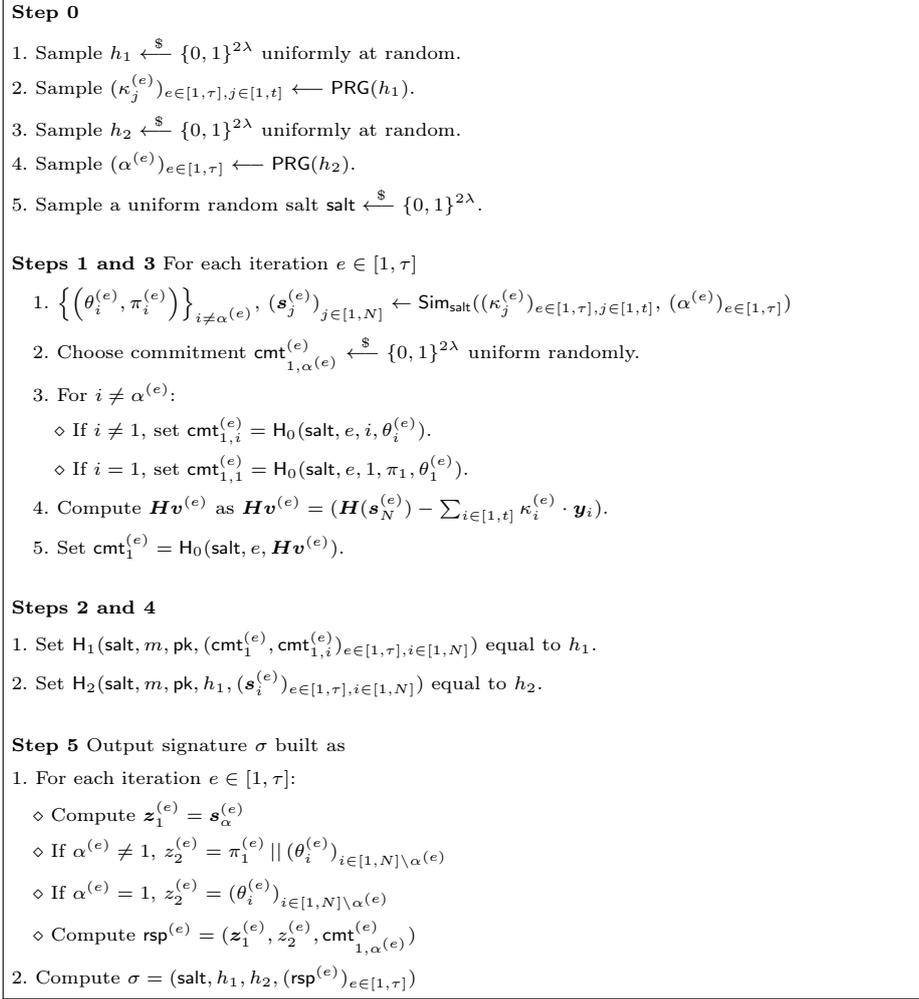


Fig. 8: Experiment 6: Answer to a signature query for a message m .

2. each $\text{cmt}_{1, i}^{(e^*)}$ in this H_1 -query was output by a previous query

$$\text{cmt}_{1, i}^{(e^*)} = \text{H}_0(\text{salt}, e^*, i, \theta_i^{(e^*)})$$

for $i \in [2, N]$, and

$$\text{cmt}_{1, 1}^{(e^*)} = \text{H}_0(\text{salt}, e^*, 1, \pi_1^{(e^*)}, \theta_1^{(e^*)})$$

for $i = 1$.

3. each $\text{cmt}_1^{(e^*)}$ in the above H_1 -query was output by a previous query

$$\text{cmt}_1^{(e^*)} = H_0 \left(\text{salt}, e^*, \left(\mathbf{H} \mathbf{s}_N^{(e^*)} - \sum_{i \in [1, t]} \kappa_i^{(e^*)} \cdot \mathbf{y}_i \right) \right)$$

4. the permutation π derived from $\{\pi_i^{(e^*)}\}_{i \in [1, N]}$ i.e. $\pi = \pi_N^{(e^*)} \circ \pi_{N-1}^{(e^*)} \circ \dots \circ \pi_1^{(e^*)}$ satisfies

$$\mathbf{H} \left(\pi \left[\sum_{i \in [1, t]} \kappa_i \cdot \mathbf{x}_i \right] \right) = \sum_{i \in [1, t]} \kappa_i \cdot \mathbf{y}_i$$

for some $\kappa \in \mathbb{F}_q^t \setminus \mathbf{0}$, where $\kappa := \{\kappa_1, \dots, \kappa_t\}$ and $\mathbf{0} \in \mathbb{F}_q^t$ is the all zero vector.

(Note that in all cases the commitments in the relevant prior H_1 -query, if it exists, must be unique since the experiment is aborted if there is ever a collision in H_0 .)

In Experiment 7, for each query of the above form made by the adversary to H_2 (where m was not previously queried to the signing oracle), check if there exists an execution e^* which defines a correct witness. We let **Solve** be the event that this occurs for some query to H_2 . Note that, if that event occurs, the $\{\pi_i^{(e^*)}\}_{i \in [1, N]}$ (which can be determined from the oracle queries of \mathcal{A}) allow the computation of solution to r-IPKP. Therefore, $\Pr_7[\text{Solve}] \leq \epsilon_{\text{r-IPKP}}$. We claim that

$$\Pr_7 \left[\text{Forge} \wedge \overline{\text{Solve}} \right] \leq q_2 \cdot \varepsilon_{\text{KS}}^\tau,$$

where $\varepsilon_{\text{KS}} = \frac{1}{N} + \frac{N-1}{N \cdot (q^t - 1)}$ is the knowledge soundness error of one execution. To see this, assume **Solve** does not occur. Then there is no execution of any H_2 -query which defines a correct witness. When considering an arbitrary execution $e \in [1, \tau]$, the attacker can only possibly generate a forgery (using this H_2 -query) if

1. \mathcal{A} guesses the first challenge $\kappa^{(e^*)} \in \mathbb{F}_q^t \setminus \mathbf{0}$, where $\kappa^{(e^*)} := \{\kappa_1^{(e^*)}, \dots, \kappa_t^{(e^*)}\}$ and $\mathbf{0} \in \mathbb{F}_q^t$ is the all zero vector.
2. or even if $\text{cmt}_1^{(e^*)} \neq H_0(\text{salt}, e^*, (\mathbf{H} \mathbf{s}_N^{(e^*)} - \sum_{i \in [1, t]} \kappa_i^{(e^*)} \cdot \mathbf{y}_i))$ the attacker guesses the second challenge α^* such that the views of all remaining $N - 1$ parties are consistent.

Thus, the overall probability with which the attacker can generate a forgery using this H_2 -query is

$$\varepsilon_{\text{KS}}^\tau = \left(\frac{1}{q^t - 1} + \left(1 - \frac{1}{q^t - 1} \right) \cdot \frac{1}{N} \right)^\tau.$$

The final bound is obtained by taking a union bound over all queries to H_2 . This concludes the proof of Theorem 3.4. \square

D Generic Attacks against Fiat-Shamir Signatures

Kales and Zaverucha in [36], showed a generic attack on the non-interactive version of 5-round PoK schemes. The attack strategy is to guess either one of the challenges (ch_1 or ch_2) correctly which permits the prover to cheat. The attacker then aims to split the work by attempting to guess the first challenge for η^* instances out of τ parallel repetitions, and tries to guess ch_2 for the remaining $(\tau - \eta^*)$ instances.

If the attacker can guess η^* challenges for the first phase correctly, then he can answer all the N possible challenges for the $\alpha^{(e)}$ for those instances. Subsequently, to successfully cheat he has to guess the remaining $(\tau - \eta^*)$ values of $\alpha^{(e)}$ correctly.

The parameter η^* allows the attacker to balance the cost for both guessing phases. The overall cost is minimized for a choice of

$$\eta^* = \arg \min_{0 \leq \eta \leq \tau} \left\{ \frac{1}{P_1(\eta, \tau, q, t)} + N^{(\tau - \eta)} \right\} \quad (7)$$

where,

$$P_1(\eta, \tau, q, t) := \sum_{j=\eta}^{\tau} \left(\frac{1}{q^t - 1} \right)^j \left(\frac{q^t - 2}{q^t - 1} \right)^{\tau - j} \binom{\tau}{j}.$$

Finally, the total cost for the attacker is thus

$$\mathcal{W}_{KZ} = \frac{1}{P_1(\eta^*, \tau, q, t)} + N^{(\tau - \eta^*)}. \quad (8)$$

In [36] the authors classify the 5-round protocols based on whether it is possible for the verifier to detect if the tuple $(\text{cmt}, \text{ch}_1, \text{rsp}_1)$ is valid or not. If the verifier can detect the validity of this tuple then the scheme is said to possess *early abort* property. The cost of the attack varies for different schemes based on whether they satisfy the early abort property or not. Our protocol and signature scheme do not possess the early abort property and hence the expected cost of attacking the PERK signature scheme proposed in this work, is given by Eq. (8).