PERK

N. Aaraj, S. Bettaieb, L. Bidoux, A. Budroni, V. Dyseryn, A. Esser,
P. Gaborit, M. Kulkarni, V. Mateu, M. Palumbi, L. Perin, J.-P. Tillich

Oxford Post-Quantum Cryptography Summit 2023

Overview

PERK is a signature scheme based on the **PER**muted **K**ernel problem

- Fiat-Shamir based signature along with a Zero-Knowledge Proof of Knowledge (ZK PoK)
- Underlying PoK built using Multi-Party Computation in the Head (MPCitH)
- Relies on the hardness of the relaxed Inhomogeneous Permuted Kernel Problem (r-IPKP)

Agenda

Signature from ZK PoK

Zero-Knowledge Proof of Knowledge (informal)

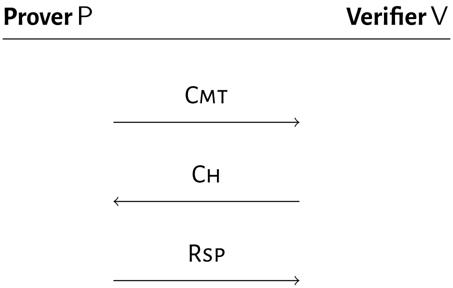


Figure 1.1: 3-rounds ZK PoK

Zero-Knowledge Proof of Knowledge (informal)

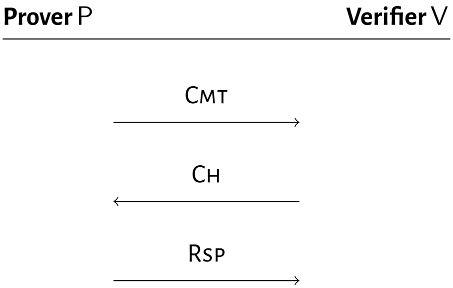


Figure 1.1: 3-rounds ZK PoK

Correctness - Honest prover P can always convince a verifier that he knows some secret s

Zero-Knowledge Proof of Knowledge (informal)

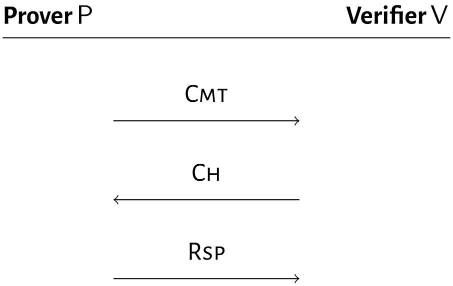


Figure 1.1: 3-rounds ZK PoK

Correctness - Honest prover P can always convince a verifier that he knows some secret s

Soundness - Malicious prover $\tilde{\mathsf{P}}$ can't convince a verifier that he knows the secret s except with negligible probability ϵ

Zero-Knowledge Proof of Knowledge (informal)

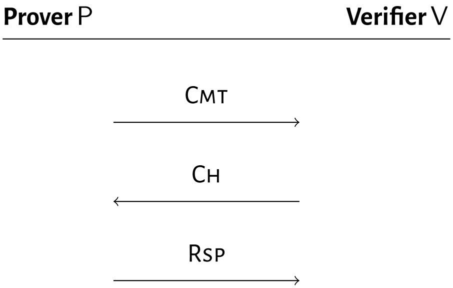


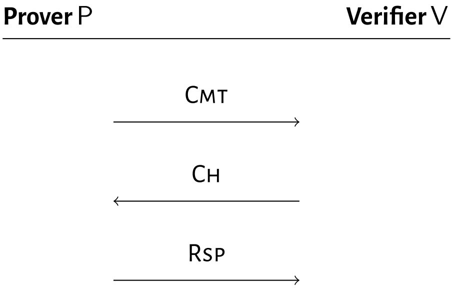
Figure 1.1: 3-rounds ZK PoK

Correctness - Honest prover P can always convince a verifier that he knows some secret s

Soundness - Malicious prover $\tilde{\mathsf{P}}$ can't convince a verifier that he knows the secret s except with negligible probability ϵ

Honest-Verifier ZK - Honest-Verifier does not learn anything on the secret s

Fiat-Shamir Transform



Objective - Transform a public coin interactive proof of knowledge into a digital signature

Fiat-Shamir Transform

Signer S

$\textsc{Cmt}$

$\textsc{Ch} = \mathcal{H}(m \,||\, \mathsf{pk} \,||\, \textsc{Cmt})$

$\textsc{Rsp}$

$$\xrightarrow{\textsc{Cmt},\ \textsc{Rsp}}$$

Figure 1.2: Fiat-Shamir Transform [FS86]

Objective - Transform a public coin interactive proof of knowledge into a digital signature

Main Idea - If the verifier V only returns strings sampled uniformly at random, it can be replaced by a hash function (modelled as random oracle)

Fiat-Shamir Transform

Signer S

$\textsc{Cmt}$

$\textsc{Ch} = \mathcal{H}(m \,||\, \mathsf{pk} \,||\, \textsc{Cmt})$

$\textsc{Rsp}$

$\xrightarrow{\textsc{Cmt},\ \textsc{Rsp}}$

Figure 1.2: Fiat-Shamir Transform [FS86]

Objective - Transform a public coin interactive proof of knowledge into a digital signature

Main Idea - If the verifier V only returns strings sampled uniformly at random, it can be replaced by a hash function (modelled as random oracle)

Security - Proven secure in the ROM for PoK using 3-rounds [PS96] and n-rounds [DGV$^+$16, AFK22]
Studied in the QROM [DFMS19, DFM20]

ZK PoK from MPC

Multi-Party Computation

Let x be a secret that can be recomputed from N shares $(\llbracket x_1 \rrbracket, \cdots, \llbracket x_N \rrbracket)$

Multi-Party Computation

Let x be a secret that can be recomputed from N shares $([\![x_1]\!], \cdots, [\![x_N]\!])$

Secure MPC [GMW87] allows a set of parties $(P_1, \cdots, P_N)$ with inputs $([\![x_1]\!], \cdots, [\![x_N]\!])$ to

- Compute $y = f(x)$ for some function f [correctness]
- Without leaking anything on x beyond what can be learned from $f(x)$ [privacy]

Multi-Party Computation

Let x be a secret that can be recomputed from N shares $(\llbracket x_1 \rrbracket, \cdots, \llbracket x_N \rrbracket)$

Secure MPC [GMW87] allows a set of parties $(P_1, \cdots, P_N)$ with inputs $(\llbracket x_1 \rrbracket, \cdots, \llbracket x_N \rrbracket)$ to

- Compute $y = f(x)$ for some function f [correctness]
- Without leaking anything on x beyond what can be learned from $f(x)$ [privacy]

For Fiat-Shamir based signature schemes, adversaries are modelled as *Honest-but-Curious*

MPC-in-the-Head Transform

Prover P	**Verifier** V

Objective - Transform a MPC protocol computing $y = f(x)$ into a ZK PoK verifying if $y = f(x)$

MPC-in-the-Head Transform

Prover P		**Verifier** V
Generates MPC shares Run MPC "in-its-Head"		
	CMT $\longrightarrow$	
		Choose a random party α
	CH $\longleftarrow$	
Reveal the shares of all parties except α and the output of α in the MPC protocol		

Objective - Transform a MPC protocol computing $y = f(x)$ into a ZK PoK verifying if $y = f(x)$

Main Idea - Prover P generates and commits to shares of x then emulates "in its head" the MPC protocol and reveals the views of $(N-1)$ parties

MPC-in-the-Head Transform

Prover P		**Verifier** V
Generates MPC shares Run MPC "in-its-Head"		
	Cmt $\longrightarrow$	
		Choose a random party α
	Ch $\longleftarrow$	
Reveal the shares of all parties except α and the output of α in the MPC protocol		
	Rsp $\longrightarrow$	
		Check commitments Check computation and result of the MPC protocol

Figure 2.1: MPC-in-the-Head [IKOS07]

Objective - Transform a MPC protocol computing $y = f(x)$ into a ZK PoK verifying if $y = f(x)$

Main Idea - Prover P generates and commits to shares of x then emulates "in its head" the MPC protocol and reveals the views of $(N-1)$ parties

Verifier V checks that the received views are consistent with commitments and checks the computation and result of the MPC protocol

MPC-in-the-Head Transform

Resulting PoK

- Correctness - From the correctness of the MPC protocol
- Zero-Knowledge - From the (N - 1)-privacy of the MPC protocol
- Soundness - Soundness error equal to $1/N$
  Can be made negligible by repeating the protocol τ times

MPC-in-the-Head Transform

Resulting PoK

- Correctness - From the correctness of the MPC protocol
- Zero-Knowledge - From the (N - 1)-privacy of the MPC protocol
- Soundness - Soundness error equal to $1/N$
  Can be made negligible by repeating the protocol τ times

Reducing the PoK size [KKW18]

- Compress commitments by hasing them together
- Compress seeds associated to each party using a Merkle tree

ZK PoK for r-IPKP

IPKP & r-IPKP

The **P**ermuted **K**ernel **P**roblem was initially introduced in [Sha90]

IPKP & r-IPKP

The **P**ermuted **K**ernel **P**roblem was initially introduced in [Sha90]

Definition (Inhomogeneous Permuted Kernel Problem)

Input - $\mathbf{H} \in \mathbb{F}_q^{m \times n}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ for $i \in [t]$ with $\mathbf{X} = (\mathbf{x}_1 \mid \cdots \mid \mathbf{x}_t)$ a full rank matrix

$\pi \in \mathcal{S}_n$ such that $\mathbf{H}\big(\pi[\mathbf{x}_i]\big) = \mathbf{y}_i$ for $i \in [t]$

Goal - Find $\tilde{\pi}$ such that $\mathbf{H}\big(\tilde{\pi}[\mathbf{x}_i]\big) = \mathbf{y}_i$ for $i \in [t]$

IPKP & r-IPKP

The **P**ermuted **K**ernel **P**roblem was initially introduced in [Sha90]

Definition (Inhomogeneous Permuted Kernel Problem)

Input - $\mathbf{H} \in \mathbb{F}_q^{m\times n}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ for $i \in [t]$ with $\mathbf{X} = (\mathbf{x}_1 \mid \cdots \mid \mathbf{x}_t)$ a full rank matrix

$\pi \in \mathcal{S}_n$ such that $\mathbf{H}\big(\pi[\mathbf{x}_i]\big) = \mathbf{y}_i$ for $i \in [t]$

Goal - Find $\tilde{\pi}$ such that $\mathbf{H}\big(\tilde{\pi}[\mathbf{x}_i]\big) = \mathbf{y}_i$ for $i \in [t]$

- Mono-dimensional IPKP [$\mathbf{t} = \mathbf{1}$]
- Multi-dimensional IPKP [$\mathbf{t} > \mathbf{1}$]

IPKP & r-IPKP

Definition (Relaxed Inhomogeneous Permuted Kernel Problem)

Input - $\mathbf{H} \in \mathbb{F}_q^{m\times n}$, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{F}_q^n \times \mathbb{F}_q^m$ for $i \in [t]$ with $\mathbf{X} = (\mathbf{x}_1 \mid \cdots \mid \mathbf{x}_t)$ a full rank matrix

$\pi \in \mathcal{S}_n$ such that $\mathbf{H}\big(\pi[\mathbf{x}_i]\big) = \mathbf{y}_i$ for $i \in [t]$

Goal - Find $\tilde{\pi}$ 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 $(\kappa_1, \ldots, \kappa_t) \in \mathbb{F}_q^t \setminus \mathbf{0}$

Known Attacks against IPKP & r-IPKP

Attacks on IPKP

- Mono-dimensional case studied in [Geo92, BCCG93, PC94, JJ01, LP11, KMP19, SBC22]
- Existing attacks generalized to the multi-dimensional case in [SBC22]

Known Attacks against IPKP & r-IPKP

Attacks on IPKP

- Mono-dimensional case studied in [Geo92, BCCG93, PC94, JJ01, LP11, KMP19, SBC22]
- Existing attacks generalized to the multi-dimensional case in [SBC22]

Attacks on r-IPKP

- Existing attacks generalized to the relaxed case in PERK

Known Attacks against IPKP & r-IPKP

Attacks on IPKP

- Mono-dimensional case studied in [Geo92, BCCG93, PC94, JJ01, LP11, KMP19, SBC22]
- Existing attacks generalized to the multi-dimensional case in [SBC22]

Attacks on r-IPKP

- Existing attacks generalized to the relaxed case in PERK

Parameter sets considered in PERK use $t = 3$ or $t = 5$

PoK for r-IPKP

PERK is derived from a 5-rounds ZK PoK introduced in [BG23]

PoK for r-IPKP

PERK is derived from a 5-rounds ZK PoK introduced in [BG23]

Overview

- PoK uses r-IPKP for challenge space amplification [BG23]
  - First challenge space has size $|\mathcal{C}_1| = q^t - 1$

PoK for r-IPKP

PERK is derived from a 5-rounds ZK PoK introduced in [BG23]

Overview

- PoK uses r-IPKP for challenge space amplification [BG23]
  - First challenge space has size $|\mathcal{C}_1| = q^t - 1$
- PoK uses shared permutation [FJR23] to compute $\pi[\mathbf{x}]$ without leaking anything on π
  - From $(\pi_i, \mathbf{v}_i)_{i \in [N]}$, compute $\pi = \pi_1 \circ \cdots \circ \pi_N$ and $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
  - Compute $\mathbf{s}_N = \pi[\mathbf{x}] + \mathbf{v}$ by recurrence from $\mathbf{s}_0 = \mathbf{x}$ and $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{Hv}$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{H}\mathbf{v}$

Verifier V_0

4. Sample $(\kappa_i)_{i \in [t]} \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{Hv}$

Verifier V_0

4. Sample $(\kappa_i)_{i \in [t]} \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$

Prover P_1

5. Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$ using $\mathbf{s}_0 = \sum_{i \in [t]} \kappa_i \cdot \mathbf{x}_i$
6. Commit to $(\mathbf{s}_i)_{i \in [N]}$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{Hv}$

Verifier V_0

4. Sample $(\kappa_i)_{i \in [t]} \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$

Prover P_1

5. Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$ using $\mathbf{s}_0 = \sum_{i \in [t]} \kappa_i \cdot \mathbf{x}_i$
6. Commit to $(\mathbf{s}_i)_{i \in [N]}$

Verifier V_1

7. Sample $\alpha \xleftarrow{\$} [1, N]$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{Hv}$

Verifier V_0

4. Sample $(\kappa_i)_{i \in [t]} \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$

Prover P_1

5. Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$ using $\mathbf{s}_0 = \sum_{i \in [t]} \kappa_i \cdot \mathbf{x}_i$
6. Commit to $(\mathbf{s}_i)_{i \in [N]}$

Verifier V_1

7. Sample $\alpha \xleftarrow{\$} [1, N]$

Prover P_2

8. Compute $\mathbf{z_1} = \mathbf{s}_\alpha$ and $z_2 = (\pi_i, \mathbf{v}_i)_{i \in [N] \setminus \alpha}$
9. Output $\mathsf{rsp} = (\mathbf{z}_1, z_2, \mathsf{com}_\alpha)$

PoK for r-IPKP (High Level Description)

Prover P_0

1. Sample shares $(\pi_i, \mathbf{v}_i)$ with $\pi_1 = \pi_2^{-1} \circ \cdots \circ \pi_N^{-1} \circ \pi$
2. Compute $\mathbf{v} = \mathbf{v}_N + \sum_{i \in [N-1]} \pi_N \circ \cdots \circ \pi_{i+1}[\mathbf{v}_i]$
3. Commit to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ and $\mathbf{Hv}$

Verifier V_0

4. Sample $(\kappa_i)_{i \in [t]} \xleftarrow{\$} \mathbb{F}_q^t \setminus \mathbf{0}$

Prover P_1

5. Compute $\mathbf{s}_i = \pi_i[\mathbf{s}_{i-1}] + \mathbf{v}_i$ using $\mathbf{s}_0 = \sum_{i \in [t]} \kappa_i \cdot \mathbf{x}_i$
6. Commit to $(\mathbf{s}_i)_{i \in [N]}$

Verifier V_1

7. Sample $\alpha \xleftarrow{\$} [1, N]$

Prover P_2

8. Compute $\mathbf{z_1} = \mathbf{s}_\alpha$ and $z_2 = (\pi_i, \mathbf{v}_i)_{i \in [N] \setminus \alpha}$
9. Output rsp $= (\mathbf{z}_1, z_2, \text{com}_\alpha)$

Verifier V_2

10. Check commitments to $(\pi_i, \mathbf{v}_i)_{i \in [N]}$ using z_2 and com_α
11. Check commitments to $(\mathbf{s}_i)_{i \in [N]}$ using $\mathbf{s}_0$ and $\mathbf{z}_1$
12. Check commitment to $\mathbf{Hv}$ using $\mathbf{Hs}_N - \sum_{i \in [t]} \kappa_i \cdot \mathbf{y}_i$

Sizes & Performances

Resulting Sizes

- $|\mathsf{sk}| = \underbrace{\lambda}_{\text{Seed for } \pi}$ $\quad |\mathsf{pk}| = \underbrace{\lambda}_{\text{Seed for } \mathbf{H} \text{ and } (\mathbf{x}_i)_{i\in[t]}} + \underbrace{t \cdot m\lceil \log_2(q) \rceil}_{\text{Vectors } (\mathbf{y}_i)_{i\in[t]} \text{ in } \mathbb{F}_q^m}$

Resulting Sizes

- $|\mathsf{sk}| = \underbrace{\lambda}_{\text{Seed for } \pi}$ $\qquad |\mathsf{pk}| = \underbrace{\lambda}_{\text{Seed for } \mathbf{H} \text{ and } (\mathbf{x}_i)_{i\in[t]}} + \underbrace{t \cdot m\lceil \log_2(q) \rceil}_{\text{Vectors } (\mathbf{y}_i)_{i\in[t]} \text{ in } \mathbb{F}_q^m}$

- $|\sigma| \approx \tau \cdot \Big(\underbrace{n\lceil \log_2(q) \rceil}_{\text{Vector } \mathbf{s}_\alpha \text{ in } \mathbb{F}_q^n} + \underbrace{n\lceil \log_2(n) \rceil}_{\text{Permutation } \pi_1} + \underbrace{\lambda\lceil \log_2(N) \rceil}_{\text{Seeds for parties } i \in [1,N] \setminus \alpha} + \underbrace{2\lambda}_{\text{Commitment for party } \alpha} \Big)$

Short Parameters

NIST level	sk	pk	σ	Keygen	Sign	Verify
PERK L1 [t = 3]	16 B	0.15 kB	6.56 kB	80 k	39 M	27 M
PERK L1 [t = 5]	16 B	0.24 kB	6.06 kB	91 k	36 M	25 M
PERK L3 [t = 3]	24 B	0.23 kB	15.0 kB	175 k	82 M	65 M
PERK L3 [t = 5]	24 B	0.37 kB	13.8 kB	194 k	77 M	60 M
PERK L5 [t = 3]	32 B	0.31 kB	26.4 kB	300 k	185 M	143 M
PERK L5 [t = 5]	32 B	0.51 kB	24.2 kB	328 k	171 M	131 M

Table 1: Sizes and performances (CPU cycles)
[Constant-Time implementation using AVX2 @3GHz]

Fast Parameters

NIST level	sk	pk	σ	Keygen	Sign	Verify
PERK L1 [t = 3]	16 B	0.15 kB	8.35 kB	77 k	7.6 M	5.3 M
PERK L1 [t = 5]	16 B	0.24 kB	8.03 kB	90 k	7.2 M	5.1 M
PERK L3 [t = 3]	24 B	0.23 kB	18.8 kB	167 k	16 M	13 M
PERK L3 [t = 5]	24 B	0.37 kB	18.0 kB	185 k	15 M	12 M
PERK L5 [t = 3]	32 B	0.31 kB	33.3 kB	304 k	36 M	28 M
PERK L5 [t = 5]	32 B	0.51 kB	31.7 kB	324 k	34 M	26 M

Table 2: Sizes and performances (CPU cycles)
[Constant-Time implementation using AVX2 @3GHz]

Advantages & Limitations

Advantages & Limitations

Advantages

- **Good public key + signature size** & small public and private keys
- Underlying **hardness assumption is unstructured**
- **Resilience against IPKP and r-IPKP attacks** - Increasing the r-IPKP parameters has a limited impact on the signature size

Advantages & Limitations

Limitations

- **Relatively slow** similarly to most MPC based schemes
- **Relatively large signature size** similarly to most MPC based schemes
- Rely on a **variant of the IPKP problem**

What's Next ?

Disclaimer - Work in progress, results are not guaranted

Expected update

- ◇ Improved signature size (approx. **-5%**)
- ◇ Improved implementation

pqc-perk.org

Thank you for your attention.

References I

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß.
Fiat-Shamir transformation of multi-round interactive proofs.
In Theory of Cryptography Conference (TCC), pages 113–142. Springer, 2022.

[BCCG93] Thierry Baritaud, Mireille Campana, Pascal Chauvaud, and Henri Gilbert.
On the Security of the Permuted Kernel Identification Scheme.
In Annual International Cryptology Conference (CRYPTO), pages 305–311. Springer, 1993.

[BG23] Loïc Bidoux and Philippe Gaborit.
Compact Post-quantum Signatures from Proofs of Knowledge Leveraging Structure for the PKP, SD and RSD Problems.
In Codes, Cryptology and Information Security (C2SI), pages 10–42. Springer, 2023.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz.
The measure-and-reprogram technique 2.0: multi-round fiat-shamir and more.
In Annual International Cryptology Conference (CRYPTO), pages 602–631. Springer, 2020.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner.
Security of the fiat-shamir transformation in the quantum random-oracle model.
In Annual International Cryptology Conference (CRYPTO), pages 356–383. Springer, 2019.

References II

[DGV+16] Özgür Dagdelen, David Galindo, Pascal Véron, Sidi Mohamed El Yousfi Alaoui, and Pierre-Louis Cayrel.
Extended security arguments for signature schemes.
Designs, Codes and Cryptography, 78(2):441–461, 2016.

[FJR23] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain.
Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature.
Designs, Codes and Cryptography, 91(2):563–608, 2023.

[FS86] Amos Fiat and Adi Shamir.
How to prove yourself: Practical solutions to identification and signature problems.
In Annual International Cryptology Conference (CRYPTO). Springer, 1986.

[Geo92] Jean Georgiades.
Some Remarks on the Security of the Identification Scheme Based on Permuted Kernels.
Journal of Cryptology, 5(2):133–137, 1992.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson.
How to play any mental game, or a completeness theorem for protocols with honest majority.
In ACM Symposium on Theory of Computing (STOC), pages 218–229, 1987.

References III

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.
Zero-Knowledge from Secure Multiparty Computation.
In ACM Symposium on Theory of Computing (STOC), pages 21–30, 2007.

[JJ01] Éliane Jaulmes and Antoine Joux.
Cryptanalysis of PKP: A new approach.
In International Conference on Practice and Theory of Public-Key Cryptography (PKC), pages 165–172. Springer, 2001.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang.
Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures.
In ACM Conference on Computer and Communications Security (CCS), 2018.

[KMP19] Eliane Koussa, Gilles Macario-Rat, and Jacques Patarin.
On the complexity of the Permuted Kernel Problem.
Cryptology ePrint Archive, Report 2019/412, 2019.

[LP11] Rodolphe Lampe and Jacques Patarin.
Analysis of some natural variants of the PKP algorithm.
Cryptology ePrint Archive, Report 2011/686, 2011.

References IV

[PC94] Jacques Patarin and Pascal Chauvaud.
Improved Algorithms for the Permuted Kernel Problem.
In Annual International Cryptology Conference (CRYPTO), pages 391–402. Springer, 1994.

[PS96] David Pointcheval and Jacques Stern.
Security proofs for signature schemes.
In International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), 1996.

[SBC22] Paolo Santini, Marco Baldi, and Franco Chiaraluce.
Computational Hardness of the Permuted Kernel and Subcode Equivalence Problems.
Cryptology ePrint Archive, Report 2022/1749, 2022.

[Sha90] Adi Shamir.
An Efficient Identification Scheme Based on Permuted Kernels.
In Annual International Cryptology Conference (CRYPTO). Springer, 1990.