# PKP-Based Signature Scheme

Ward Beullens[1], Jean-Charles Faugère[2], Eliane Koussa[3], Gilles Macario-Rat[4], Jacques Patarin[5], and Ludovic Perret[2]

[1] imec-COSIC, KU Leuven ward.beullens@esat.kuleuven.be
[2] INRIA and Sorbonne Universities/UPMC Uni Paris 6
jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr
[3] Versailles Laboratory of Mathematics, UVSQ EJKoussa@outlook.com
[4] Orange gilles.macariorat@orange.com
[5] Versailles Laboratory of Mathematics, UVSQ, CNRS, University of Paris-Saclay, jpatarin@club-internet.fr

**Abstract.** In this document, we introduce PKP-DSS: a Digital Signature Scheme based on the Permuted Kernel Problem(PKP) [21]. PKP is a simple NP-hard [10] combinatorial problem that consists of finding a kernel for a publicly known matrix, such that the kernel vector is a permutation of a publicly known vector. This problem was used to develop an Identification Scheme (IDS) which has a very efficient implementation on low-cost smart cards. From this zero-knowledge identification scheme we derive PKP-DSS with the traditional Fiat-Shamir transform [9]. Thus, PKP-DSS has a security that can be provably reduced, in the *(classical) random oracle model*, to the hardness of random instances of PKP (or, if wanted, to any specific family of PKP instances). We propose parameter sets following the thorough analysis of the State-of-the-art attacks on PKP presented in [**17**]. We show that PKP-DSS is competitive with other signatures derived from Zero-Knowledge identification schemes. In particular, PKP-DSS-128 gives a signature size of approximately 20 KBytes for 128 bits of classical security, which is approximately 30% smaller than MQDSS. Moreover, our proof-of-concept implementation shows that PKP-DSS-128 is an order of magnitude faster than MQDSS which in its turn is faster than Picnic2, SPHINCS,...
Since the PKP is NP hard and since there are no known quantum attacks for solving PKP significantly better than classical attacks, we believe that our scheme is post-quantum secure.

**Keywords:** public-key cryptography, post-quantum cryptography, Fiat-Shamir, 5-pass identification scheme, Public-key Signature, Permuted Kernel Problem.

## 1 Introduction

The construction of large quantum computers would break all public-key cryptographic schemes in use today, because they rely on the discrete logarithm

problem or the integer factorization problem. Despite the fact that it isn't clear when large scale quantum computation would be feasible, it is important to anticipate quantum computing and design new public key cryptosytems that are resistant to quantum attacks.

Therefore, there currently is a large research effort to develop new post-quantum secure schemes, and a Post-Quantum Cryptography standardization process has been initiated by the American National Institute of Standards and Technology (`https://www.nist.gov/`). Because of this, there has been renewed interest in constructing signature scheme by applying the Fiat-Shamir transform [9] to Zero-Knowledge Identification Schemes.

In particular, we are interested in post-quantum cryptographic schemes whose security relies on the quantum hardness of some NP-Hard problem [2]. One of those problems is the Permuted Kernel Problem: the problem of finding a permutation of a known vector such that the resulting vector is in the kernel of a given matrix. This is a classical NP-Hard combinatorial problem which requires only simple operations such as basic linear algebra, and permuting the entries of a vector. For quite some time, no new attacks on PKP have been discovered, which makes it possible to confidently estimate the concrete hardness of the problem. In 1989, Shamir [21] introduced a ZK-Identification scheme, based on this Permuted Kernel Problem.

**Previous work and State-of-the-art.** Since quantum computers are expected not to be capable of solving *NP*-Hard problems in subexponential time (in worst case), Zero-knowledge Identification schemes based on such problems are interesting candidates for Post-Quantum Cryptography. The Fiat-Shamir transform [9] is a technique that can convert such a zero knowledge authentication scheme into a signature scheme.

This approach was taken by Chen et al. [7], who applied the Fiat-Shamir transform to a 5-pass identifiation scheme of Sakumoto et al. [20]. This identification scheme relies on the hardness of the (*NP*-Hard) problem of finding a solution to a set of multivariate quadratic equations. Chen et al. proved that, in the random oracle model, applying the Fiat-Shamir transform to this 5-pass identification scheme results in a secure signature scheme. A concrete parameter choice and an efficient implementation of this signature scheme (which is called MQDSS) was developed, and this was one of the submissions to the NIST PQC standardization project. At a security level of 128 bits, the MQDSS scheme comes with a public key of 46 Bytes, a secret key of 16 Bytes and a signature size of approximately 16.2 Kilobytes.

A different line of work resulted in the Picnic signature scheme. Chase et al. [6] constructed this digital signature scheme by applying the Fiat-Shamir transform to an identification scheme whose security relies purely on symmetric primitives. At the 128 bit security level Picnic has a public key of 32 Bytes, a

secret key of 16 Bytes and signatures of approximately 33 Kilobytes. There is a second version of this signature scheme, where the signatures are only 13.5 Kilobytes, but Picnic2 is 45 times slower than the original Picnic for signing and 25 times slower for verification.

**Main results.** The main contribution of this paper is to present PKP-DSS, a new post-quantum secure signature scheme. Similar to the approaches cited above, we use the Fiat-Shamir transform to construct a signature scheme from the 5-pass PKP identification scheme by Shamir [21]. Following the complexity analysis of the PKP [17], we choose secure parameter sets for the signature scheme for multiple security levels. Moreover we have developed a constant time C implementation of the new signature scheme. By constant-time we mean that the running time and the memory access pattern of the implementation are independent of secret material, therefore blocking attacks from timing side channels.

The resulting signature scheme compares well with MQDSS and Picnic/Picnic2. Our scheme is much faster than MQDSS and Picnic/Picnic2 in terms of signing and verification, we have small public and private keys, and the signature sizes of our scheme are comparable to those of MQDSS and Picnic2. This makes our signature scheme based on PKP competitive with state of the art post-quantum signature schemes.

## 2 The Permuted Kernel Problem

In order to introduce the signature scheme, we first present the PKP problem [21]. We also briefly present the best technique for solving it. In [11], Georgiades presents symmetric polynomials equations which will be utilized by all the other attacks. The authors of [1] investigate also the security of PKP, where a time-memory trade-off was introduced. Moreover, Patarin and Chauvaud improve algorithms for the Permuted Kernel Problem[19]. Also, in [14], a new time-memory trade-off was proposed. After all, it appears in [17] that the attack of Patarin and Chauvaud [19] is the most efficient.

### 2.1 Introduction to PKP

PKP [21,10] is the problem on which the security of PKP-DSS is based. PKP is a linear algebra problem which asks to find a kernel vector of a given matrix under a vector-entries constraint. It's a generalization of the Partition problem [10, pg.224]. More precisely, it is defined as follows:

**Definition 1 (Permuted Kernel Problem).** *Given a finite field $\mathbb{F}_p$, a matrix $\mathbf{A} \in \mathbb{F}_p^{m \times n}$ and a $n$-vector $\mathbf{v} \in \mathbb{F}_p^n$, find a permutation $\pi \in S_n$ such that $\mathbf{A}\mathbf{v}_\pi = 0$, where $\mathbf{v}_\pi = (v_{\pi(1)}, \cdots, v_{\pi(n)})$*

A reduction of the 3-Partition problem proves PKP to be NP-Hard [10]. Moreover, solving random instances of PKP seems hard in practice. In fact, this is the fundamental design assumption of PKP-DSS. The hardness of PKP comes from, on the one hand, the big number of permutations, on the other hand, from the small number of possible permutations which satisfy the kernel equations. More precisely, PKP is hard because it obligates the choice of a vector, with already fixed set of entries, from the kernel of the matrix $\mathbf{A}$. Note that, to make the problem more difficult, it is desirable that the $n$-vector $\mathbf{v}$ has distinct coordinates, otherwise if there are repeated entries, the space of permutations of $\mathbf{v}$ gets smaller. In the next section, we give the best known algorithm to solve the PKP problem.

### 2.2 The algorithm of Patarin-Chauvaud

The implementation's efficiency of the first IDS, proposed by Shamir [21], based on PKP problem has led to several solving tools. In fact, there are various attacks for PKP, which are all exponential. We will not describe them here, instead we refer to [17] for further details .

Patarin and Chauvaud combine in [19] the two ideas presented in the previous attacks [11,1]. The result was a reduction in the time required to attack PKP. They also present some new ideas in order to reduce this time the memory needed. Thus, this leads to a new algorithm which is quicker and more efficient than all the given attacks of PKP [11,1,15]. The details and the numerical results are given in the main article [17].

### 2.3 Commitment schemes

In our protocol, we use a commitment scheme $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^\star \to \{0,1\}^{2\lambda}$, that takes as input $\lambda$ uniformly random bits $\mathsf{bits}$, where $\lambda$ is the security parameter, and a message $m \in \{0,1\}^\star$ and outputs a $2\lambda$ bit long commitment $\mathsf{Com}(\mathsf{bits}, m)$. In the description of our protocols, we often do not explicitly mention the commitment randomness. We write $\mathsf{S} \leftarrow \mathsf{Com}(m)$, to denote the process of picking a uniformly random bit string $\mathsf{r}$, and setting $\mathsf{C} \leftarrow \mathsf{Com}(\mathsf{r}, m)$. Similarly, when we write check $\mathsf{C} = \mathsf{Com}(m)$, we actually mean that the prover communicates $\mathsf{r}$ to the verifier, and that the verifier checks if $\mathsf{C} = \mathsf{Com}(\mathsf{r}, m)$.

We assume that $\mathsf{Com}$ is computationally binding, which means that no computationally bounded adversary can produce a $\mathsf{r}, \mathsf{r}', m, m'$ with $m \neq m'$ such that $\mathsf{Com}(\mathsf{r}, m) = \mathsf{Com}(\mathsf{r}', m')$. We also assume that $\mathsf{Com}$ is computationally hiding, which means that for every pair of mesages $m, m'$, no computationally bounded adversary can distinguish the distributions of $\mathsf{Com}(m) = \mathsf{Com}(m')$.

## 3 Identification scheme (IDS) based on PKP

In this section, we present the 5-pass Zero-Knowledge Identification Scheme (ZK-IDS) based on the computational hardness of PKP [21,18], noted here PKP-IDS.

We first quote and refer to some of the general definitions given in [7] : Identification scheme, Completeness, Soundness (with soundness error), Honest-verifier zero-knowledge, and also in [13,8] : computationally hiding commitment, computationally binding commitment. We then apply and adapt these definitions to the Identification scheme based on PKP and give and prove its own properties of performance and security. This approach will be more convenient for presenting the signature scheme in the next section.

### 3.1 Preliminaries

In what follows and as in [7], we assume the existence of a non-interactive commitment scheme Com which is computationally hiding and computationally binding (see [13,8] for details). The commitments are computed using the function Com. For notational convenience, we do not explicitly write down the commitment randomness. In fact, as will be mentioned later, we do not use commitment randomness in the implementation of the signature.

### 3.2 PKP 5-pass IDS

In this section, we present (a slightly modified version of) PKP-IDS. It consists of three probabilistic polynomial time algorithms $IDS = (\textsc{KeyGen}, \mathcal{P}, \mathcal{V})$ which we will describe now.

**Generation of the public key and secret key in PKP-IDS.** The users first agree on a prime number $p$, and on $n, m$, the dimensions of the matrix $\mathbf{A}$. The public-key in PKP-IDS is an instance of PKP a solution to this instance is the secret-key. Thus, the prover picks a (right) kernel-vector $\mathbf{w} \in \mathrm{Ker(A)}$, then randomly generates a secret permutation of $n$ elements $\mathsf{sk} = \pi$ and finishes by computing $\mathbf{v} = \mathbf{w}_{\pi^{-1}}$. We summarize the key generation algorithm in Alg. 1.

**5-pass identification protocol: Prover $\mathcal{P}$ and Verifier $\mathcal{V}$.**
The prover and verifier are interactive algorithms that realize the identification protocol in 5 passes. The 5 passes consist of one commitment and two responses transmitted from the prover to the verifier and two challenges transmitted from the verifier to the prover. The identification protocol is summarized in Algorithm 3.2.

From Shamir in [21] we have the following results.

**Algorithm 1** KeyGen

$\mathbf{A} \xleftarrow{\$} \mathbb{F}_p^{m \times n}$
$\mathbf{w} \xleftarrow{\$} \mathrm{Ker}(\mathbf{A})$
$\pi \xleftarrow{\$} S_n$
$\mathbf{v} \leftarrow \mathbf{w}_{\pi^{-1}}$
**Return** $(\mathsf{pk} = (\mathbf{A}, \mathbf{v}), \mathsf{sk} = \pi)$

**Algorithm 2** The 5 pass PKP identification protocol

| $\mathcal{P}(\mathsf{sk}, \mathsf{pk})$ | | $\mathcal{V}(\mathsf{pk})$ |
|---|---|---|
| $\sigma \xleftarrow{\$} S_n$ | | |
| $\mathbf{r} \xleftarrow{\$} \mathbb{F}_p^n$ | | |
| $\mathsf{C}_0 \leftarrow \mathrm{Com}(\sigma, \mathbf{Ar})$ | | |
| $\mathsf{C}_1 \leftarrow \mathrm{Com}(\pi\sigma, \mathbf{r}_\sigma)$ | | |
| | $\xrightarrow{\mathsf{C}_0, \mathsf{C}_1}$ | |
| | | $c \xleftarrow{\$} \mathbb{F}_p$ |
| | $\xleftarrow{c}$ | |
| $\mathbf{z} \leftarrow \mathbf{r}_\sigma + c\mathbf{v}_{\pi\sigma}$ | | |
| | $\xrightarrow{\mathbf{z}}$ | |
| | | $b \xleftarrow{\$} \{0, 1\}$ |
| | $\xleftarrow{b}$ | |
| **if** $b = 0$ **then** | | |
| $\quad \mathsf{resp} \leftarrow \sigma$ | | |
| **else** | | |
| $\quad \mathsf{resp} \leftarrow \pi\sigma$ | | |
| **end if** | | |
| | $\xrightarrow{\mathsf{resp}}$ | |
| | | **if** $b = 0$ **then** |
| | | $\quad$ accept if $\mathsf{C}_0 = \mathrm{Com}(\sigma, A\mathbf{z}_{\sigma^{-1}})$ |
| | | **else** |
| | | $\quad$ accept if $\mathsf{C}_1 = \mathrm{Com}(\pi\sigma, \mathbf{z} - c\mathbf{v}_{\pi\sigma})$ |
| | | **end if** |

**Theorem 1.** *PKP-IDS is complete, moreover if the used commitment scheme is computationally hiding then PKP-IDS is computationally honest-verifier zero-knowledge and if the commitment scheme is computationally binding, then PKP-IDS is sound with soundness error* $\kappa = \frac{p+1}{2p}$.

The soundness error comes from the fact that the prover can easily cheat if he can predict either $c$ (which he can do with probability $1/p$) or $b$ (which he can do with probability $1/2$). The probability of guessing either $c$ or $b$ is then $\frac{p+1}{2p}$. For example, a prover cheater without knowledge of $\pi$ can always respond with $\mathbf{z} = \mathbf{r}_\sigma$, in the case $b = 1$ he just responds with a random permutation. In this case the prover will accept the proof if $c = 0$ or $b = 0$. The soundness proof says that for any cheating prover, the success probability is at most $\frac{p+1}{2p} + \epsilon$, where $\epsilon$ is a negligible function of the security parameter. To get the cheating probability down, we repeat the protocol several times. Repeating the zero-knowledge proof $N$ times results in an Identification scheme with knowlegde error

$$\kappa = \left(\frac{p+1}{2p}\right)^N ,$$

hence it suffices to repeat the protocol $\lceil \lambda / \log_2(\frac{2p}{p+1}) \rceil$ times to get a soundness error $\kappa \leq 2^{-\lambda}$.

### 3.3 Optimizations

We now describe a number of optimizations to reduce the communication cost of the identification scheme, as well as the computational cost of the algorithms. We will start by explaining a few standard optimizations that are common for identification protocols based on zero knowledge proofs. Then, we will explain some novel optimizations that apply to the specific context of PKP-IDS.

**Hashing the commitments.** In the commitment phase of the protocol, instead of transmitting all the $2N$ commitments $\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}$ the prover can just hash all these commitments together with a collision resistant hash function $\mathcal{H}$ and only transmit the hash $h = \mathcal{H}(\mathsf{C}_0^{(1)}, \cdots, \mathsf{C}_1^{(N)})$. Then, the prover includes the $N$ commitments $\mathsf{C}_{1-b_i}^{(i)}$ in the second response. Since the verifier can reconstruct the $\mathsf{C}_{b_i}^{i}$ himself, he now has all the $2N$ commitments, so he can hash them together and check if their hash matched $h$. With this optimization we reduce the number of communicated commitments from $2N$ to $N$, at the cost of transmitting a single hash value.

**Use seeds and PRG.** Instead of directly choosing the permutation $\sigma$ at random, we can instead choose a random seed of $\lambda$ bits and use a PRG to expand this seed into a permutation $\sigma$. This way, instead of transmitting $\sigma$, we can just transmit the $\lambda$-bit seed. This reduces the communication cost per permutation from $\log_2(n!)$ bits to just $\lambda$ bits. For example for 128-bits of security, we have $n = 69$, so the communication cost per permutation drops from $\log_2(69!) \approx 327$ bits to just 128 bits.

**Matrix A in systematic form.** Now we get to the PKP-IDS-specific optimizations. With high probability, we can perform elementary row operations on $\mathbf{A}$ to put it in the form $\begin{pmatrix} I_m & \mathbf{A}' \end{pmatrix}$, for some $(n-m)$-by-$m$ matrix $\mathbf{A}'$. Since row operations do not affect the right kernel of $A$, we can just choose the matrix $\mathbf{A}$ of this form during key generation, without affecting the security of the scheme. This makes the protocol more efficient, because multiplying by a matrix of this form requires only $(n-m) * m$ multiplications instead of $n * m$ multiplications for a general matrix multiplication.

**Optimizing key generation.** It is of course not very efficient to include the matrix $\mathbf{A}$ in the public key, because this is a large matrix. The first idea is to just pick a random seed, and use a PRG to expand this seed to obtain the matrix $\mathbf{A}$. The public key then consists of a random seed, and the vector $\mathbf{v}$ of length $n$. However, we can do slightly better than this. We can use a seed to generate the first $n-1$ columns of $\mathbf{A}$ and the vector $v$. Then we pick a random permutation $\pi$, and we solve for the last column of $\mathbf{A}$ such that $\mathbf{v}_\pi$ is in the right kernel of $\mathbf{A}$. Now the public key only consists of a seed and a vector of length $m$ (instead of a vector of length $n$). Another important advantage of this approach is that we do not need to do Gaussian elimination this way (and if fact this was the motivation behind this optimization).

**Use seeds and PRG again.** Because of the second optimization, we can send a $\lambda$-bit seed instead of $\sigma$, if the challenge bit $b = 0$. However, in the case $b = 1$, we still need to send the permutation $\pi\sigma$, because we cannot generate both $\sigma$ and $\pi\sigma$ with a PRG. However, this problem can be solved. We can generate $\mathbf{r}_\sigma$ with a PRG, and then we can send this seed instead of $\pi\sigma$. This seed can be used to compute $\pi\sigma$, because if the verifier knows $\mathbf{z}$ and $\mathbf{r}_\sigma$, then he can compute $\mathbf{z} - \mathbf{r}_\sigma = c\mathbf{v}_{\pi\sigma}$. And since $\mathbf{v}$ and $c$ are known, it is easy to recover $\pi\sigma$ from $c\mathbf{v}_{\pi\sigma}$ (we choose the parameters such that the entries of $v$ are all distinct, so there is a unique permutation that maps $\mathbf{v}$ to $\mathbf{v}_{\pi\sigma}$). Morever, sending the seed for $\mathbf{r}_\sigma$ does not reveal more information than sending $\pi\sigma$, because given $\mathbf{z}$ and $\pi\sigma$ it is trivial to compute $\mathbf{r}_\sigma$, so this optimization does not affect the security of the scheme. However, there is a problem: If $c = 0$, then the $c\mathbf{v}_{\pi\sigma} = 0$, and so the verifier cannot recover $\pi\sigma$. To solve this problem we just restrict the challenge

space to $\mathbb{F}_p \setminus \{0\}$. This increases the soundness error to $\frac{p}{2p-2}$ (instead of $\frac{p+1}{2p}$), but this is not a big problem. An important advantage of this optimization is that the signature size is now constant. Without this optimization, a response to the challenge $b = 0$ would be smaller than a response to $b = 1$. But with the optimization the second response is always a random seed, regardless of the value of $b$.

### 3.4 Communication cost

We can now provide the communication complexity of $N$ rounds of the IDS, of which the soundness error is $\left(\frac{p}{2p-2}\right)^N$. The commitment consists of a single hash value, which is only $2\lambda$ bits. The first response consists of $N$ vectors of length $n$ over $\mathbb{F}_p$, so this costs $Nn\lceil \log_2 p \rceil$ bits of communication. Lastly, the second responses consist of $N$ random $\lambda$-bit seeds, $N$ commitments (which consist of $2\lambda$ bits each) and $N$ commitment random strings (which consist of $\lambda$ bits each), so this costs $4N\lambda$ bits of communcation. In total, the communication cost (ignoring the challenges) is

$$2\lambda + N\left(n\lceil \log_2 p \rceil + 4\lambda\right) .$$

## 4 Digital signature scheme (DSS) based on PKP

We present here the main contribution of this work which is to construct a digital signature scheme, based on the PKP problem, from the optimized IDS defined in Section 3. This is simply a direct application of the well-known Fiat Shamir transformation [9].

The key generation algorithm is identical to the key generation algorithm for the identification scheme. To sign a message $m$, the signer executes the first phase of the commitment scheme to get a commitment $\mathsf{com}$. Then he derives the first challenge $\mathbf{c} = (c_1, \cdots, c_N)$ from $m$ and $\mathsf{com}$ by evaluating a hash function $\mathcal{H}_1(m||\mathsf{com})$. Then he does the next phase of the identification protocol to get the $N$ response vectors $\mathsf{rsp_1} = (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$. Then he uses a second hash function to derive $\mathbf{b} = (b_1, \ldots, b_N)$ from $m$, $\mathsf{com}$ and $\mathsf{resp_1}$ as $\mathcal{H}_2(m||\mathsf{com}, \mathsf{resp_1})$. Then he finishes the identification protocol to obtain the vector of second responses $\mathsf{resp_2} = (\mathsf{resp}^{(1)}, \cdots, \mathsf{resp}^{(N)})$. Then, the signature is simply $(\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$.

To verify a signature $(\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$ for a message $m$, the verifier simply uses the hash function $\mathcal{H}_1$ and $\mathcal{H}_2$ to obtain $\mathbf{c}$ and $\mathbf{b}$ respectively. Then, he verifies that $(\mathsf{com}, \mathbf{c}, \mathsf{resp}_1, \mathbf{b}, \mathsf{resp}_2)$ is a valid transcript of the identification protocol.

The signing and verification algorithms are displayed in Algorithm 3 and 4 in more detail.

We then get the same security result as Th. 5.1 in [7].

**Algorithm 3** Sign($\mathsf{sk}, m$)

derive $\mathbf{A}, \mathbf{v}$ and $\pi$ from $\mathsf{sk}$.
**for** $i$ from 1 to $N$ **do**
  pick $\lambda$-bit seeds $\mathsf{seed}_0^{(i)}$ and $\mathsf{seed}_1^{(i)}$ uniformly at random
  $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}_0^{(i)})$
  $\mathbf{r}_\sigma^{(i)} \leftarrow \mathsf{PRG}_2(\mathsf{seed}_1^{(i)})$
  $\mathsf{C}_0^{(i)} = \mathrm{Com}\big(\sigma^{(i)}, \mathbf{A}\mathbf{r}^{(i)}\big)$,
  $\mathsf{C}_1^{(i)} = \mathrm{Com}\big(\pi\sigma^{(i)}, \mathbf{r}_\sigma^{(i)}\big)$.
**end for**
$\mathsf{com} := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}\big)$
$c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1\big(m||\mathsf{com}\big)$.  $c^i \in \mathbb{F}_p \setminus \{0\}$
**for** $i$ from 1 to $N$ **do**
  $\mathbf{z}^{(i)} \leftarrow \mathbf{r}_\sigma^{(i)} + c^{(i)}\mathbf{v}_{\pi\sigma^{(i)}}$
**end for**
$\mathsf{resp}_1 \leftarrow (\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)})$
$b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{resp}_1)$
**for** $i$ from 1 to $N$ **do**
  $\mathsf{resp}_2^{(i)} \leftarrow (\mathsf{seed}_{b^{(i)}}^{(i)} || \mathsf{C}_{1-b^{(i)}}^{(i)})$
**end for**
$\mathsf{resp}_2 \leftarrow (\mathsf{resp_2}^{(1)}, \cdots, \mathsf{resp_2}^{(N)})$
**Return** $(\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$

**Algorithm 4** Verify($m, \mathsf{pk}, \sigma = (\mathsf{com}, \mathsf{resp}_1, \mathsf{resp}_2)$)

$c^{(1)}, \cdots, c^{(N)} \leftarrow \mathcal{H}_1\big(m||\mathsf{com}\big)$.
$b^{(1)}, \cdots, b^{(N)} \leftarrow \mathcal{H}_2(m||\mathsf{com}||\mathsf{resp}_1)$
Parse $\mathsf{resp}_1$ as $\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(N)}$
Parse $\mathsf{resp}_2$ as $\mathsf{seed}^{(1)}, \cdots, \mathsf{seed}^{(N)}, \mathsf{C}_{1-b^{(1)}}^{(1)}, \cdots, \mathsf{C}_{1-b^{(N)}}^{(N)}$
**for** $i$ from 1 to $N$ **do**
  **if** $b^{(i)} = 0$ **then**
    $\sigma^{(i)} \leftarrow \mathsf{PRG}_1(\mathsf{seed}^{(i)})$
    $\mathsf{C}_0^{(i)} \leftarrow \mathrm{Com}(\sigma^{(i)}, \mathbf{A}\mathbf{z}_{\sigma^{(i)}-1})$
  **else**
    $\mathbf{r}_\sigma^{(i)} \leftarrow \mathsf{PRG}_2(\mathsf{seed}^{(i)})$
    **if** $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$ is not a permutation of $c\mathbf{v}$ **then**
      **Return** reject
    **else**
      $\pi\sigma^{(i)} \leftarrow$ the permutation that maps $c\mathbf{v}$ to $\mathbf{z}^{(i)} - \mathbf{r}_\sigma$.
    **end if**
    $\mathsf{C}_1^{(i)} \leftarrow \mathrm{Com}(\pi\sigma^{(i)}, \mathbf{r}_\sigma^{(i)})$
  **end if**
**end for**
$\mathsf{com}' := \mathcal{H}_{\mathsf{com}}\big(\mathsf{C}_0^{(1)}, \mathsf{C}_1^{(1)}, \cdots, \mathsf{C}_0^{(N)}, \mathsf{C}_1^{(N)}\big)$
**Return** accept if and only if $\mathsf{com} = \mathsf{com}'$

**Theorem 2.** *PKP-DSS is Existential-Unforgeable under Chosen Adaptive Message Attacks (EU-CMA) in the random oracle model, if*

- *the search version of the Permuted Kernel problem is intractable,*
- *the hash functions and pseudo-random generators are modeled as random oracles,*
- *the commitment functions are computationally binding, computationally hiding, and the probability that their output takes a given value is negligible in the security parameter.*

The proof is the same as in [7].

### 4.1 Generic attack

If the number of iterations $N$ is chosen such that $(\frac{p}{2p-2})^N \leq 2^{-\lambda}$, then the cheating probability of the identification protocol is bounded by $2^{-\lambda}$. However, a recent attack by Kales and Zaverucha on MQDSS reveals that this does not meant that the Fiat-Shamir signature scheme has $\lambda$ bits of security [16]. They give a generic attack that also applies to PKP-DSS. The attack exploits the fact that if an attacker can guess the first challenge **or** the second challenge, he can produce responses that the verifier will accept. The idea is to split up the attack in two phases. In the first phase, the attacker guesses the values of the $N$ first challenges, and uses this guess to produce commitments. Then, he derives the challenges from the commitment and he hopes that at least $k$ of his $N$ guesses are correct. This requires on average

$$\mathsf{Cost}_1(N,k) = \sum_{i=k}^{N} \left(\frac{1}{p-1}\right)^k \left(\frac{p-2}{p-1}\right)^{N-k} \binom{N}{k}$$

trials. In the second phase, the attacker guesses the values of second challenges, and uses these guesses to generate a response. Then he derives the second challenges with a hash function and he hopes that his guess was correct for the $N-k$ rounds of the identification protocol where he did not guess the first challenge correctly. This requires on average $2^{N-k}$ tries. Therefore, the total cost of the attack is

$$\min_{0\leq k\leq N} \mathsf{Cost}_1(N,k) + 2^{N-k}\,.$$

## 5 Parameter choice and Implementation

### 5.1 Parameter choice

The choice for $p$ was to select prime numbers close to powers of 2, such as 251, 509 and 4093. Then, $n$ and $m$ were related by the formula $p^m \approx n!$. And

finally, using the time complexity of Poupard's algorithm, triplets of $(p, n, m)$ were selected matching the security requirements and optimizing the size of the signature. With these parameter choices the scheme is secure against the attacks described in [17]. We pick the value of $N$ just large enough such that

$$\min_{0 \leq k \leq N} \mathsf{Cost}_1(N, k) + 2^{N-k} \geq 2^{\lambda} ,$$

such that the scheme is secure against the generic attack of Kales and Zaverucha [16]. The chosen parameter sets for three different security levels are shown in Table 1. security levels are shown in Table 1.

| Parameter Set | Security level | $p$ | $n$ | $m$ | Iterations $N$ | Attack cost |
|---|---|---|---|---|---|---|
| PKP-DSS-128 | 128 | 251 | 69 | 41 | 157 | $2^{130}$ |
| PKP-DSS-192 | 192 | 509 | 94 | 54 | 229 | $2^{193}$ |
| PKP-DSS-256 | 256 | 4093 | 106 | 47 | 289 | $2^{257}$ |

**Table 1.** PKP-DSS Parameters sets

### 5.2 Key and signature sizes

**Public key.** A public key consists of the last column of $\mathbf{A}$ and a random seed, which is used to generate all but the last column of $\mathbf{A}$ and the vector $\mathbf{v}$. Therefore, the public key consist of $\lambda + m\lfloor \log_2(p) \rfloor$ bits.

**Secret key.** A secret key is just a random seed that was used to seed the key generation algorithm, therefore it consists of only $\lambda$ bits.

**Signature.** Finally, a signature consists of a transcript of the identification protocol (excluding the challenges, because they are computed with a hash function). In Sect 3.4 we calculated that a transcript can be represented with $2\lambda + N\left(n\lceil \log_2 p \rceil + 3\lambda\right)$ bits, so this is also the signature size.

In Table 2 we summarize the key and signature sizes for the parameter sets proposed in the previous section.

### 5.3 Implementation

To showcase the efficiency of PKP-DSS and to compare the performance to existing Fiat-Shamir signatures we made a proof-of-concept implementation in plain C. The code of our implementation is available on github at [4]. We have used

| Security level | Parameters $(p, n, m, N)$ | \|sk\| Bytes | \|pk\| Bytes | \|sig\| Kilobytes |
|---|---|---|---|---|
| 128 | $(251, 69, 41, 157)$ | 16 | 57 | 20.4 |
| 192 | $(509, 94, 54, 229)$ | 24 | 85 | 43.4 |
| 256 | $(4093, 106, 47, 289)$ | 32 | 103 | 76.5 |

**Table 2.** Key and signature sizes for PKP-DSS with the three proposed parameter sets.

SHA-3 as hash function and commitment scheme, and we have used SHAKE128 as extendable output function. The running time of the signing and verification algorithms is dominated by expanding seeds into random vectors and random permutations. This can be sped up by using a vectorized implementation of SHAKE128, and using vector instructions to convert the random bitstring into a vector over $\mathbb{F}_p$ or a permutation in $S_n$. We leave this task for future work.

**Making the implementation constant time.** Most of the key generation and signing algorithms is inherently constant time (signing branches on the value of the challenge bits $b$, but this does not leak information because $b$ is public). The only problem was that applying a secret permutation to the entries of a vector, when implemented naively, involves accessing data at secret indices. To prevent this potential timing leak we used the "djbsort" constant time sorting code [3]. More specifically, we combine the permutation and the vector into a single list of $n$ integers, where the permutation is stored in the most significant bits, and the entries of the vector are stored in the least significant bits. Then we sort this list of integers in constant time and we extract the permuted vector from the low order bits. Relative to the naive implementation this slows down signing by only 11%. There is no significant slowdown for key generation.

### 5.4 Performance results

To measure the performance of our implementation we ran experiments on a laptop with a i5-8250U CPU running at 1.8 GHz. The C code was compiled with gcc version 7.4.0 with the compile option `-O3`. The cycle counts in Table 3 are averages of 10000 key generations, signings and verifications.

### 5.5 Comparison with existing FS signatures

In Table 4, we compare PKP-DSS to MQDSS, Picnic, and Picnic2. We can see that for all the schemes the public and secret keys are all very small. The main differences are signature size and speed. When compared to MQDSS, the

| Security level | Parameters $(p, n, m, N)$ | KeyGen $10^3$ cycles | Sign $10^3$ cycles | Verify $10^3$ cycles |
|---|---|---|---|---|
| 128 | $(251, 69, 41, 157)$ | 72 | 2518 | 896 |
| 192 | $(509, 94, 54, 229)$ | 121 | 5486 | 2088 |
| 256 | $(4093, 106, 47, 289)$ | 151 | 7411 | 3491 |

**Table 3.** Average cycle counts for key generation, signing and verification, for our implementation of PKP-DSS with the three proposed parameter sets.

signature sizes of PKP-DSS are roughly 30% smaller, while being a factor 14 and 30 faster for signing and verification respectively. Compared to Picnic, the PKP-DSS signatures are roughly 40% smaller, and signing and verification is 4 and 9 times faster respectively. Compared to Picinc2 our scheme is 153 and 170 times faster for signing and verification, but this comes at the cos of signatures which are 50% larger. Finally, compared to SUSHSYFISH [12], a different scheme based on the Permuted Kernel Problem, our scheme is 3.4 and 6.6 times faster, but at the cost of signatures that are 45% larger.

| Security level | Scheme | Secret key (Bytes) | Public key (Bytes) | Signature (KBytes) | Sign $10^6$ cycles | Verify $10^6$ cycles |
|---|---|---|---|---|---|---|
| 128 | PKP-DSS-128 | 16 | 57 | 20.4 | 2.5 | 0.9 |
| | MQDSS-31-48 | 16 | 46 | 28.0 | 36 | 27 |
| | Picnic-L1-FS | 16 | 32 | 33.2 | 10 | 8.4 |
| | Picnic2-L1-FS | 16 | 32 | 13.5 | 384 | 153 |
| | SUSHSYFISH-1 | 16 | 72 | 14.0 | 8.6 | 6 |
| 192 | PKP-DSS-192 | 24 | 85 | 43.4 | 5.5 | 2.1 |
| | MQDSS-31-64 | 24 | 64 | 58.6 | 116 | 85 |
| | Picnic-L3-FS | 24 | 48 | 74.9 | 24 | 20 |
| | Picnic2-L3-FS | 24 | 48 | 29.1 | 1183 | 357 |
| | SUSHSYFISH-3 | 24 | 108 | 30.8 | 22.7 | 16.5 |
| 256 | PKP-DSS-256 | 32 | 103 | 76.5 | 7.4 | 3.5 |
| | Picnic-L5-FS | 32 | 64 | 129.7 | 44 | 38 |
| | Picnic2-L5-FS | 32 | 64 | 53.5 | 2551 | 643 |
| | SUSHSYFISH-5 | 32 | 142 | 54.9 | 25.7 | 18 |

**Table 4.** Comparison of different post-quantum Fiat-Shamir schemes

## 6 Conclusion

We introduces a new post-quantum secure signature scheme PKP-DSS, which is based on a PKP Zero-knowledge identification scheme [21]. We optimized this

identification scheme, and to make it non-interactive, we used the well-known Fiat-Shamir transform.

We developped a constant time implementation of PKP-DSS and we conclude that our scheme is competitive with other Post-Quantum Fiat-Shamir signature schemes such as MQDSS, Picnic/Picnic2 and SUSHSYFISH. The main advantages of our scheme are that signing and verification are much faster than existing FS signatures, and that the scheme is very simple to implement. Our implementation takes only 600 lines of C code, including comments and empty lines.

## References


1. Baritaud, T., Campana, M., Chauvaud, P., Gilbert, H. On the security of the permuted kernel identification scheme. In Annual International Cryptology Conference (pp. 305-311). (1992, August), Springer, Berlin, Heidelberg.
2. Bennett, C. H., Bernstein, E., Brassard, G., Vazirani, U. (1997). Strengths and weaknesses of quantum computing. SIAM journal on Computing, 26(5), 1510-1523.
3. The djbsort software library for sorting arrays of integers or floating-point numbers in constant time. `https://sorting.cr.yp.to/`
4. Beullens, Ward. PKPDSS, (2019) Public GitHub repository. `https://github.com/WardBeullens/PKPDSS`
5. Beullens, Ward. On sigma protocols with helper for MQ and PKP, fishy signature schemes and more. Cryptology ePrint Archive, Report 2019/490, 2019. `https://eprint.iacr.org/2019/490`.
6. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., ... Zaverucha, G. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1825-1842) (2017, October). ACM.
7. Chen, M. S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P. MQDSS specifications. (2018).
8. Damgård, I. Commitment schemes and zero-knowledge protocols. In School organized by the European Educational Forum (pp. 63-86). (1998, June). Springer, Berlin, Heidelberg.
9. Fiat, A., Shamir, A. (1986, August). How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology—CRYPTO'86 (pp. 186-194). Springer, Berlin, Heidelberg.
10. Gary, M., Johnson, D. (1979). Computers and Intractability: A Guide to NP-Completeness. New York: W H.
11. Georgiades, J. (1992). Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology, 5(2), 133-137.
12. W. Beullens. "On sigma protocols with helper for MQ and PKP, fishy signature schemes and more" IACR Cryptology ePrint Archive 2019 (2019): 490.
13. Haitner, I., Nguyen, M. H., Ong, S. J., Reingold, O., Vadhan, S. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM Journal on Computing, 39(3), pp. 1153-1218.

14. Jaulmes, É., Joux, A. Cryptanalysis of pkp: a new approach. In International Workshop on Public Key Cryptography (pp. 165-172). (2001, February) Springer, Berlin, Heidelberg.
15. Joux, A., Lercier, R. "Chinese and Match", an alternative to Atkin's "Match and Sort" method used in the SEA algorithm. Mathematics of computation, 70(234), 827-836.
16. Daniel Kales, Greg Zaverucha. "Forgery attacks against MQDSSv2.0" Note postes on the NIST PQC forum `https://groups.google.com/a/list.nist.gov/forum/?utm_medium=email&utm_source=footer#!msg/pqc-forum/LlHhfwg73eQ/omM6TWwlEwAJ`
17. Koussa, Eliane, Gilles Macario-Rat, and Jacques Patarin. "On the complexity of the Permuted Kernel Problem." IACR Cryptology ePrint Archive 2019 (2019): 412.
18. Lampe, R., Patarin, J. Analysis of Some Natural Variants of the PKP Algorithm. IACR Cryptology ePrint Archive, 2011, 686.
19. Patarin, J., Chauvaud, P. Improved algorithms for the permuted kernel problem. In Annual International Cryptology Conference (pp. 391-402). (1993, August) Springer, Berlin, Heidelberg.
20. Sakumoto, K., Shirai, T., Hiwatari, H. (2011, August). Public-key identification schemes based on multivariate quadratic polynomials. In Annual Cryptology Conference (pp. 706-723). Springer, Berlin, Heidelberg.
21. Shamir, A. (1989, August). An efficient identification scheme based on permuted kernels. In Conference on the Theory and Application of Cryptology (pp. 606-609). Springer, New York, NY.