On sigma protocols with helper for MQ and PKP, fishy signature schemes and more

Ward Beullens[1]

imec-COSIC KU Leuven,
Kasteelpark Arenberg 10 - bus 2452, 3001 Heverlee, Belgium
Ward.Beullens@esat.kuleuven.be

Abstract. This work presents 2 sigma protocols with helper to prove knowledge of:

- A solution to a system of quadratic polynomials
- A solution to an instance of the Permuted Kernel Problem

We then remove the helper from the protocol with a "cut-and-choose" protocol and we apply the Fiat-Shamir transform to obtain signature schemes with security proof in the QROM. We show that the resulting signature schemes, which we call the "MUltivarite quaDratic FIat-SHamir" scheme (MUDFISH) and the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH), are more efficient than existing signatures based on the MQ problem and the Permuted Kernel Problem. We also leverage the ZK-proof for PKP to improve the efficiency of Stern-like Zero Knowledge proofs for lattice statements.

Keywords: Zero-Knowledge, Post-Quantum digital signatures, Multivariate cryptography, Permuted Kernel Problem, Silly acronyms

1 Introduction

One way to construct a signature scheme is to first construct an identification scheme and then make it into a non-interactive signature scheme with a transformation such as the the Fiat-Shamir transform [8] or the Unruh transform [19]. Looking at the NIST Post-Quantum Standardization project, two of the Round II signature schemes, MQDSS and Picnic use this approach. MQDSS [5] uses an identification scheme based on the hardness of solving a a large system of polynomial equations over a finite field. Picnic [4] uses an identification scheme constructed using the "MPC-in-the-head" technique [10] that relies on symmetric primitives. A third example is PKP-DSS [7], which uses an old identification scheme based on the hardness of the Permuted Kernel Problem (PKP) [18]. A drawback of these schemes is that the underlying identification schemes usually have a large soundness error, so a large number of parallel repetitions is

required to get a secure signature scheme. This increases the signature sizes and the signing and verification times.

Recently, Katz et al. [13] improved on the approach of Picnic by building a zero knowledge proof from MPC in the preprocessing model, where the parties can use some auxiliary data that was generated during a preprocessing phase. The advantage of moving to the new MPC protocol is that it allows for secure computation with dishonest majority with an arbitrary number of parties n, which results in a zero knowledge proof with a soundness error of $\frac{1}{n}$. Hence, fewer parallel rounds are required to get a secure signature scheme. A "cut-and-choose" protocol is used to deal with the preprocessing phase, which makes signing and verification slower compared to the original Picnic scheme.

Contributions. In this paper we generalize the approach of [13] to something we call "sigma protocols with helper". This means that at the beginning of each execution of the protocol there is a helper party that honestly generates some auxiliary information that he sends to the verifier. The helper also sends the seed that he used to generate his randomness to the prover. Then the protocol resumes like a normal sigma protocol.

We then construct sigma protocols with helper to prove knowledge of a solution of a system of quadratic equations or knowledge of a solution of a PKP instance. Our proofs have soundness error $\frac{1}{q}$, where q is the size of the finite fields that are used, which is much better than existing proofs which have soundness error $\frac{1}{2} + \frac{1}{2q}$. We then show how to remove the helper with a "cut-and-choose" protocol, analogous to the approach used by Katz et al. [13]. This transformation gives rise to zero knowledge protocols (without helper) which can then be transformed into signature schemes using the Fiat-Shamir transform.

Our proof-of-concept implementation shows that the resulting signature schemes, which we call the "MUltivarite quaDratic FIat-SHami" scheme (MUDFISH) and the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH), have smaller signatures and are faster than the existing schemes based on the MQ problem and the permuted kernel problem (i.e. MQDSS and PKP-DSS respectively). For NIST PQC Security Level 1 the MUDFISH signatures are 11.2 KB large, which is 46% smaller than the MQDSS signatures. At the same time our proof-of-concept implementation is almost twice as fast as the reference implementation of MQDSS. The task of implementing an AVX2-optimized implementation of MUDFISH and to compare to the AVX2 optimized implementation of MQDSS is left for future work. The SUSHSYFISH signatures are 30% smaller than those of PKP-DSS while our proof-of-concept implementation is slightly faster than the implementation from [7]. Moreover, unlike MQDSS and PKP-DSS, the MUDFISH and SUSHSYFISH signature schemes are based on 3-round zero knowledge proofs, hence the results of Don et al. [6] can prove

that our signature schemes are secure in the QROM. A comparison of the signature sizes and signing speed of MUDFISH and multiple instantiations of SUSHSYFISH with those of existing Post-Quantum Fiat-Shamir signatures is given in Fig. 1. Our implementation is available on GitHub [2].

Finally, since the (inhomogeneous) SIS problem embeds efficiently into the (inhomogeneous) PKP problem with the decomposition-extension technique of ling et al. [14], we observe that our ZK proof for PKP can be used as a more efficient ZK proof for the SIS (and ISIS) relations. With our proof system one can prove that a Ring-LWE ciphertext is a valid encryption of a plaintext which is known to the prover with a proof size of 405 KB. In contrast, if one was to use Stern's protocol we estimate that the proof size would be 16 times larger.

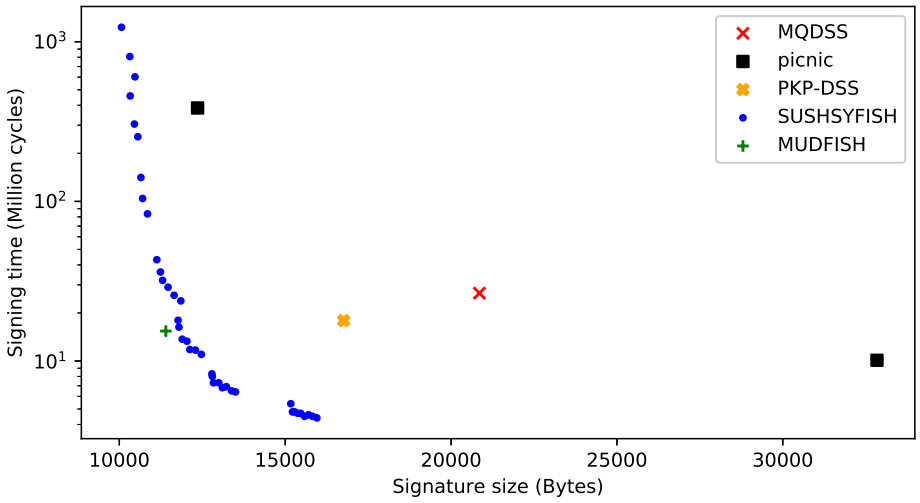

Fig. 1. Comparison of MUDFISH and SUSHSYFISH to existing Fiat-Shamir signatures. Cycle counts of picnic and MQDSS are taken from the NIST Round2 submission packages (the optimized, but not AVX2 optimized implementations), cycle counts for PKP-DSS are taken from [7].

Roadmap In Sect. 2 we formalize the notion of a sigma protocol with helper, then we construct sigma protocols with helper for the MQ problem and the Permuted Kernel Problem in sections 3 and 4. In Sect. 5 we show how to convert a sigma protocol with helper in a normal zero knowledge proof (without helper). Then, we convert our zero knowledge proofs into signature schemes in Sect. 7, where we also briefly discuss our proof-of-concept implementations. Finally we

show how to use the PKP proof to construct a zero-knowledge proof for the ISIS relation.

2 Sigma protocols with helper

This paper introduces two Sigma protocols with helper, which are like normal sigma protocols, with the addition of a trusted third party (called the helper) that runs a setup algorithm based on a random seed at the beginning of each execution of the protocol. The helper then sends some auxiliary information to the verifier and sends the seed value that was used to seed the setup algorithm to the prover. A more formal definition is as follows:

Definition 1 (Sigma protocol with helper). *A protocol is a sigma protocol with helper for relation R with challenge space $\mathcal{C}$ if it is of the form of Fig. 2 and satisfies:*

- **Completeness** *If all parties (Helper, Prover and Verifier) follow the protocol on input $(x, w) \in R$, then the verifier always accepts.*
- **2-Special soundness.** *From any two valid transcripts $(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$ and $(x, \mathsf{aux}, \mathsf{com}, ch', \mathsf{rsp}')$ with $ch \neq ch'$ and where $\mathsf{aux} = Setup(\mathsf{seed})$ for some seed value seed one can efficiently compute a witness w such that $(x, w) \in R$.*
- **Special honest-verifier zero knowledge.** *There exists a PPT simulator $\mathcal{S}$ that on input x, a random seed value seed and a random challenge ch outputs a transcript $(x, \mathsf{aux}, \mathsf{com}, ch, \mathsf{rsp})$ with $\mathsf{aux} = Setup(\mathsf{seed})$ that is computationally indistinguishable from the probability distribution of transcripts of honest executions of the protocol on input (x, w) for some w such that $(x, w) \in R$, conditioned on the auxiliary information being equal to aux and the challenge being equal to ch.*

3 Proving knowledge of a solution to a system of quadratic equations

Two zero knowledge proofs to prove knowledge of a solution of a system of multivariate quadratic equations over a finite field $\mathbb{F}_q$ were proposed by Sakumoto et al. [17]. The first proof is a 3 round protocol which has soundness error $\frac{2}{3}$, while the second proof is a five round protocol with soundness error $\frac{1}{2} + \frac{1}{2q}$, where q is the size of the finite field over which the system of polynomials is defined. The MQDSS [5] signature scheme is obtained by applying the Fiat-Shamir transform

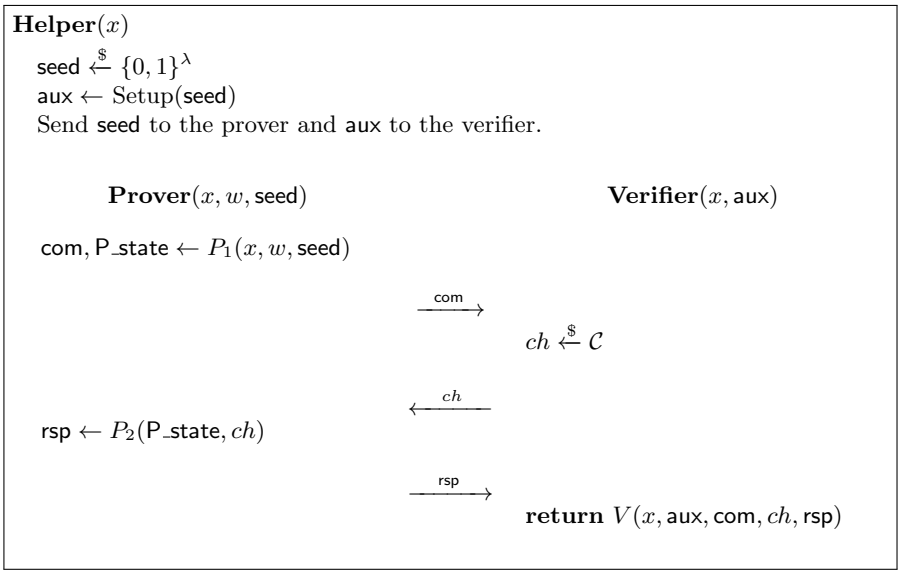

Fig. 2. The structure of a sigma protocol with trusted setup.

to the 5 round protocol of Sakumoto et al. Because the soundness error of $\frac{1}{2} + \frac{1}{2q}$ is rather big, a large number (e.g. 135 for the NIST security level I parameter set) of parallel rounds is required to obtain a secure signature scheme.

In this section we present a sigma protocol with helper to prove knowledge of a solution of a system of multivariate quadratic equations. The scheme improves the knowledge error to only $1/q$, but this comes at the cost of having an honest party that helps the prover and the verifier in their execution of the protocol. Similar to the schemes of Sakumoto et al. the new protocol relies on the fact that if $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is a multivariate quadratic map of m polynomials in n variables, then the polar form of $\mathcal{F}$, which is defined as

$$\mathcal{G}(\mathbf{x}, \mathbf{y}) \coloneqq \mathcal{F}(\mathbf{x} + \mathbf{y}) - \mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{y}) \tag{1}$$

is linear in both $\mathbf{x}$ and $\mathbf{y}$.

To prove knowledge of a secret $\mathbf{s}$ such that $\mathcal{F}(\mathbf{s}) = \mathbf{v}$ the protocol goes as follows: During the first phase the helper picks a random vector $\mathbf{r_0}$ and commits to linear secret sharings $\mathbf{t} + \mathbf{t}_c = c\mathbf{r}_0, \mathbf{e} + \mathbf{e}_c = c\mathcal{F}(\mathbf{r}_0)$ for each $c \in \mathbb{F}_q$. These commitments are public auxiliary information which the helper sends to the verifier. The helper also sends the seed that he used to generate his randomness to the prover. Then, the prover publishes the masked secret $\mathbf{r}_1 = \mathbf{s} - \mathbf{r}_0$ and commits to the value of $\mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t})$. Finally the verifier challenges the prover to reveal $\mathbf{e}_\alpha$ and $\mathbf{t}_\alpha$ for a random choice of $\alpha \in \mathbb{F}_q$ and checks whether the following equation, which is equivalent to Eqn. 1, holds.

$$\mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t}) = c\left(\mathcal{F}(\mathbf{s}) - \mathcal{F}(\mathbf{r}_1)\right) - \mathbf{e}_c - \mathcal{G}(\mathbf{r}_1, \mathbf{t}_c), \quad \forall c \in \mathbb{F}_q \tag{2}$$

A more detailed version of the protocol is displayed in Fig. 3.

Theorem 1. *Suppose the used commitment scheme is binding and hiding, then the protocol of Fig. 3 is a sigma protocol with trusted setup as in definition 1 with challenge space* $\mathbb{F}_q$.

Proof. We prove completeness, 2-special soundness and special honest-verifier zero knowledge separately:

Completeness: The fact that in a honest execution of the protocol $\mathbf{x} = \mathbf{e} + \mathcal{G}(\mathbf{r}_1, \mathbf{t})$ follows from Eqn. 2, so completeness follows immediately.

2-Special Soundness: Suppose $\mathsf{aux} = \mathrm{Setup}(\mathsf{seed})$ and $(\mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha, (\mathbf{r}_1, \mathbf{e}_\alpha, \mathbf{t}_\alpha))$, $(\mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha, (\mathbf{r}'_1, \mathbf{e}_{\alpha'}, \mathbf{t}_{\alpha'}))$ are two transcripts with $\alpha \neq \alpha'$ that are accepted by the verifier.

Let $\mathbf{x} \coloneqq \alpha(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) - \mathbf{e}_\alpha - \mathcal{G}(\mathbf{r}_1, \mathbf{t}_\alpha)$ and $\mathbf{x}' \coloneqq \alpha'(\mathbf{v} - \mathcal{F}(\mathbf{r}'_1)) - \mathbf{e}_{\alpha'} - \mathcal{G}(\mathbf{r}'_1, \mathbf{t}_{\alpha'})$, then the verifier only accepts if we have $\mathsf{com} = \mathrm{Commit}(\mathbf{r}_1, \mathbf{x}) = \mathrm{Commit}(\mathbf{r}'_1, \mathbf{x}')$, so the binding property of Commit implies that $\mathbf{r}_1 = \mathbf{r}'_1$ and $\mathbf{x} = \mathbf{x}'$. Similarly the verifier only accepts if $\mathsf{com}_\alpha = \mathrm{Commit}(\mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $\mathsf{com}_{\alpha'} = \mathrm{Commit}(\mathbf{e}_{\alpha'}, \mathbf{t}_{\alpha'})$, which implies that $\mathbf{e} + \mathbf{e}_\alpha = \alpha\mathcal{F}(\mathbf{r_0})$, $\mathbf{t} + \mathbf{t}_\alpha = \alpha\mathbf{r_0}$, $\mathbf{e} + \mathbf{e}_{\alpha'} = \alpha'\mathcal{F}(\mathbf{r_0})$ and $\mathbf{t} + \mathbf{t}_{\alpha'} = \alpha'\mathbf{r_0}$, where $\mathbf{r}_0, \mathbf{e}$ and $\mathbf{t}$ are the values generated from seed. Putting everything together we get

$$\alpha(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) + \mathbf{e} - \alpha\mathcal{F}(\mathbf{r}_0) - \mathcal{G}(\mathbf{r}_1, \alpha\mathbf{r}_0 - \mathbf{t}) = \alpha'(\mathbf{v} - \mathcal{F}(\mathbf{r}_1)) + \mathbf{e} - \alpha'\mathcal{F}(\mathbf{r}_0) - \mathcal{G}(\mathbf{r}_1, \alpha'\mathbf{r}_0 - \mathbf{t})$$

which simplifies to

$$\begin{aligned}(\alpha - \alpha')\left(\mathcal{F}(\mathbf{r}_1) + \mathcal{F}(\mathbf{r}_0 - \mathbf{v}) + \mathcal{G}(\mathbf{r}_0, \mathbf{r_1})\right) = \\ (\alpha - \alpha')\left(\mathcal{F}(\mathbf{r}_0 + \mathbf{r}_1) - \mathbf{v})\right) = 0\,,\end{aligned}$$

so $\mathbf{r}_0 + \mathbf{r}_1 = \frac{\mathbf{t}_\alpha - \mathbf{t}_{\alpha'}}{\alpha - \alpha'} + \mathbf{r}_1$ is a solution to $\mathcal{F}(\mathbf{x}) = \mathbf{v}$.

Special honest-verifier zero knowledge: Define a simulator $\mathcal{S}$, that on input $\mathbf{v}$, a random seed value seed and a random challenge $\alpha \in \mathbb{F}_q$ does the following things:

1. recompute $\mathsf{aux}, \mathbf{e}_\alpha$ and $\mathbf{t}_\alpha$ from seed.
2. pick a uniformly random vector $\mathbf{u} \in \mathbb{F}_q^n$.
3. compute $f_{\alpha, \mathbf{e}_\alpha, \mathbf{t}_\alpha}(\mathbf{u})$, where $f_{\alpha, \mathbf{e}_\alpha, \mathbf{t}_\alpha}(\mathbf{x}) \coloneqq \alpha(\mathbf{v} - \mathcal{F}(\mathbf{x})) - \mathbf{e}_\alpha - \mathcal{G}(\mathbf{x}, \mathbf{t}_\alpha)$.
4. produce a commitment com' to $(\mathbf{u}, f_{\alpha, \mathbf{e}_\alpha, \mathbf{t}_\alpha}(\mathbf{u}))$.
5. output $(\mathbf{v}, \mathsf{aux}, \mathsf{com}', \alpha, (\mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha))$.

Then the Simulator is identical to an honest prover, except for step 2, where the honest prover sets $\mathbf{u}$ equal to $\mathbf{s}-\mathbf{r}_0$ rather than a uniformly random value. It is clear that $(\mathbf{v}, \alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $(\mathbf{v}, \alpha, \mathbf{s}-\mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ are both uniformly distributed in $\{\mathbf{v}\} \times \mathbb{F}_q \times \left(\mathbb{F}_q^n\right)^3$ and hence follow the same distribution. Since com and com_α are functions of $(\mathbf{v}, \alpha, \mathbf{s}-\mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ it follows that $(\mathbf{v}, \mathsf{com}_\alpha, \mathsf{com}', \alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $(\mathbf{v}, \mathsf{com}_\alpha, \mathsf{com}, \alpha, \mathbf{s}-\mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ also follow the same distribution. Finally, since the commitments $\mathsf{com}_{c\neq\alpha}$ are never opened, it follows from the hiding property of the commitment scheme that $(\mathbf{v}, \mathsf{aux}, \mathsf{com}', \alpha, \mathbf{u}, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ and $(\mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha, \mathbf{s}-\mathbf{r}_0, \mathbf{e}_\alpha, \mathbf{t}_\alpha)$ are indistinguishable.

Helper$(\mathcal{F})$

$\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
Generate $\mathbf{e} \in \mathbb{F}_q^m$ and $\mathbf{t}, \mathbf{r}_0 \in \mathbb{F}_q^n$ from seed.
for each c in $\mathbb{F}_q$ **do**
  $\mathbf{e}_c \leftarrow c\mathcal{F}(\mathbf{r_0}) - \mathbf{e}$
  $\mathbf{t}_c \leftarrow c\mathbf{r_0} - \mathbf{t}$
  $\mathsf{com}_c \leftarrow \mathrm{Commit}((\mathbf{e}_c, \mathbf{t}_c))$
end for
$\mathsf{aux} \leftarrow [\mathsf{com}_c |$ **for** $c \in \mathbb{F}_q]$
Send seed to the prover and aux to the verifier.

Prover$(\mathcal{F}, \mathbf{s}, \mathsf{seed})$		**Verifier**$(\mathcal{F}, \mathbf{v}, \mathsf{aux})$
Regenerate $\mathbf{e}, \mathbf{t}, \mathbf{r}_0$ from seed.		
$\mathbf{r}_1 \leftarrow \mathbf{s} - \mathbf{r}_0$		
$\mathsf{com} \leftarrow \mathrm{Commit}((\mathbf{r}_1, \mathbf{e}+\mathcal{G}(\mathbf{r}_1, \mathbf{t})))$		
	$\xrightarrow{\mathsf{com}}$	
		$\alpha \xleftarrow{\$} \mathbb{F}_q$
	$\xleftarrow{\alpha}$	
Recompute $\mathbf{e}_\alpha, \mathbf{t}_\alpha$ from seed.		
	$\xrightarrow{(\mathbf{r}_1, \mathbf{e}_\alpha, \mathbf{t}_\alpha)}$	
		$\mathbf{x} \leftarrow \alpha(\mathbf{v}-\mathcal{F}(\mathbf{r}_1))-\mathbf{e}_\alpha-\mathcal{G}(\mathbf{r}_1, \mathbf{t}_\alpha)$
		$b_1 \leftarrow \mathsf{com} = \mathrm{Commit}(\mathbf{r}_1, \mathbf{x})$
		$b_2 \leftarrow \mathsf{com}_\alpha = \mathrm{Commit}((\mathbf{e}_\alpha, \mathbf{t}_\alpha))$
		return $b_1 \wedge b_2$

Fig. 3. A sigma protocol with helper for proving knowledge of a solution to the MQ problem.

4 Proving knowledge of a solution to a PKP instance

The permuted kernel problem is an NP-hard problem that asks to, given a matrix $\mathbf{A} \in \mathbb{F}_p^{m \times n}$ and a vector $\mathbf{v} \in \mathbb{F}_p^n$, find a permutation $\pi \in S_n$ such that $\mathbf{v}_\pi$, the vector obtained by permuting the entries of $\mathbf{v}$ according to the permutation π, is in the right kernel of $\mathbf{A}$. Several works have investigated the hardness of the PKP problem, but the problem remains hard for small parameters [15, 1, 9, 11]. Shamir constructed a 5-round zero knowledge proof to prove knowledge of a solution to a PKP instance with soundness error $\frac{1}{2} + \frac{1}{2p}$ [18]. This identification scheme was turned into a signature scheme with the Fiat-Shamir transform [7].

In this section we give a Sigma protocol with helper with challenge space $\mathbb{F}_p$ to prove knowledge of a solution for a PKP instance. To prove knowledge of a solution π to the instance $(\mathbf{A}, \mathbf{v})$ the protocol goes as follows: The helper picks a random vector $\mathbf{r} \in \mathbb{F}_p^n$, and a random permutation $\sigma \in S_n$, it then commits to $\mathbf{r} + c\mathbf{v}_\sigma$ for all values of $c \in \mathbb{F}_p$. The helper sends these commitments as public auxiliary information to the verifier, and he sends the seed that he used to generate his randomness to the prover. Then the prover sends $\rho = \pi\sigma^{-1}$ to the verifier and commits to the value of $\mathbf{A}\mathbf{r}_{\pi\sigma^{-1}}$. Finally, the verifier challenges the prover to reveal $\mathbf{x} = \mathbf{r} + \alpha\mathbf{v}_\sigma$ for a random choice of α. Once the prover reveals $\mathbf{x}$ the verifier checks if $\mathbf{A}\mathbf{x}_\rho = \mathbf{A}\left(\mathbf{r}_{\pi\sigma^{-1}} + \alpha\mathbf{v}_\pi\right) = \mathbf{A}\mathbf{r}_{\pi\sigma^{-1}}$. For a more detailed description of the protocol we refer to Fig. 4.

Theorem 2. *Suppose the used commitment scheme is binding and hiding, then the protocol of Fig. 4 is a sigma protocol with trusted setup as in definition 1 with challenge space $\mathbb{F}_p$.*

Proof. We prove completeness, 2-special soundness and special honest-verifier zero knowledge separately:

Completeness: In an honest execution of the protocol the value $\mathbf{y} = \mathbf{A}\mathbf{x}_\rho$ is equal to $\mathbf{A}\left(\mathbf{r}_{\pi\sigma^{-1}} + \alpha\mathbf{v}_\pi\right)$, so if π was a solution to the PKP instance $(\mathbf{A}, \mathbf{v})$, then $\mathbf{y} = \mathbf{A}\mathbf{r}_{\pi\sigma^{-1}}$ and hence the completeness follows from the completeness of the commitment scheme.

2-Special Soundness: Suppose $\mathsf{aux} = \text{Setup}(\mathsf{seed})$ and $(\mathbf{A}, \mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha, (\rho, \mathbf{x}))$, $(\mathbf{A}, \mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha', (\rho', \mathbf{x}'))$ are two transcripts with $\alpha \neq \alpha'$ that are accepted by the verifier.

Let $\mathbf{y} \coloneqq \mathbf{A}\mathbf{x}_\rho$ and $\mathbf{y}' \coloneqq \mathbf{A}\mathbf{x}'_{\rho'}$, then the verifier only accepts if we have $\mathsf{com} = \text{Commit}(\rho, \mathbf{y}) = \text{Commit}(\rho', \mathbf{y}')$, so the binding property of Commit implies that $\rho = \rho'$ and $\mathbf{y} = \mathbf{y}'$. Similarly the verifier only accepts if $\mathsf{com}_\alpha = \text{Commit}(\mathbf{x})$ and $\mathsf{com}_{\alpha'} = \text{Commit}(\mathbf{x}')$, which implies that $\mathbf{x} = \mathbf{r} + \alpha\mathbf{v}_\sigma$ and $\mathbf{x}' = \mathbf{r} + \alpha'\mathbf{v}_\sigma$, where

$\mathbf{r}, \sigma$ are the values generated from seed during setup. Putting everything together we get

$$\mathbf{A}\left(\mathbf{r}_\rho + \alpha \mathbf{v}_{\rho\sigma}\right) = \mathbf{A}\left(\mathbf{r}_\rho + \alpha' \mathbf{v}_{\rho\sigma}\right)$$

which simplifies to

$$(\alpha - \alpha')\mathbf{A}\mathbf{v}_{\rho\sigma} = 0,$$

so $\rho\sigma$ is a solution to the Permuted Kernel Problem. The value of ρ is known to the extractor, and the value of sigma can be deduced from $\alpha, \alpha', \mathbf{x}, \mathbf{x}'$ and $\mathbf{v}$, because $\mathbf{x} - \mathbf{x}' = (\alpha - \alpha')\mathbf{v}_\sigma$. (If the entries of $\mathbf{v}$ are not unique, multiple values of σ are possible, but they will all give valid solutions to the PKP problem.)

Special honest-verifier zero knowledge: Define a simulator $\mathcal{S}$, that on input $\mathbf{A}, \mathbf{v}$, a random seed value seed and a random challenge $\alpha \in \mathbb{F}_p$ does the following things:

1. recompute aux and $\mathbf{x} = \mathbf{r} + \alpha \mathbf{v}_\sigma$ from seed.
2. pick a uniformly random permutation $\tau \in S_n$.
3. produce a commitment com' to $(\tau, \mathbf{A}\mathbf{x}_\tau)$.
4. output $(\mathbf{A}, \mathbf{v}, \mathsf{aux}, \mathsf{com}', \alpha, (\tau, \mathbf{A}\mathbf{x}_\tau))$.

Then the Simulator is identical to an honest prover, except for step 2, where the honest prover sets ρ equal to $\pi\sigma^{-1}$ rather than a uniformly random value. It is clear that $(\mathbf{A}, \mathbf{v}, \alpha, \tau, \mathbf{A}\mathbf{x}_\tau)$ and $(\mathbf{A}, \mathbf{v}, \alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ are both uniformly distributed in $\{(\mathbf{A}, \mathbf{v})\} \times \mathbb{F}_q \times S_n \times \mathbb{F}_q^n$ and hence follow the same distribution. Since com and com_α are functions of $(\mathbf{A}, \mathbf{v}, \alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ it follows that $(\mathbf{A}, \mathbf{v}, \mathsf{com}_\alpha, \mathsf{com}', \alpha, \tau, \mathbf{A}\mathbf{x}_\tau)$ and $(\mathbf{A}, \mathbf{v}, \mathsf{com}_\alpha, \mathsf{com}, \alpha, \rho, \mathbf{A}\mathbf{x}_\rho)$ also follow the same distribution. Finally, since the commitments $\mathsf{com}_{c \neq \alpha}$ are never opened, it follows from the hiding property of the commitment scheme that $(\mathbf{v}, \mathsf{aux}, \mathsf{com}', \alpha, (\tau, \mathbf{A}\mathbf{x}_\tau))$ and $(\mathbf{v}, \mathsf{aux}, \mathsf{com}, \alpha, (\rho, \mathbf{A}\mathbf{x}_\rho))$ are indistinguishable.

5 Removing the helper

In this section we show how to transform a Sigma protocol with helper into a standard zero knowledge proof of knowledge (without helper). We use the same "Cut-and-choose" approach that was used by Katz et al. to get rid of the preprocessing phase [13].

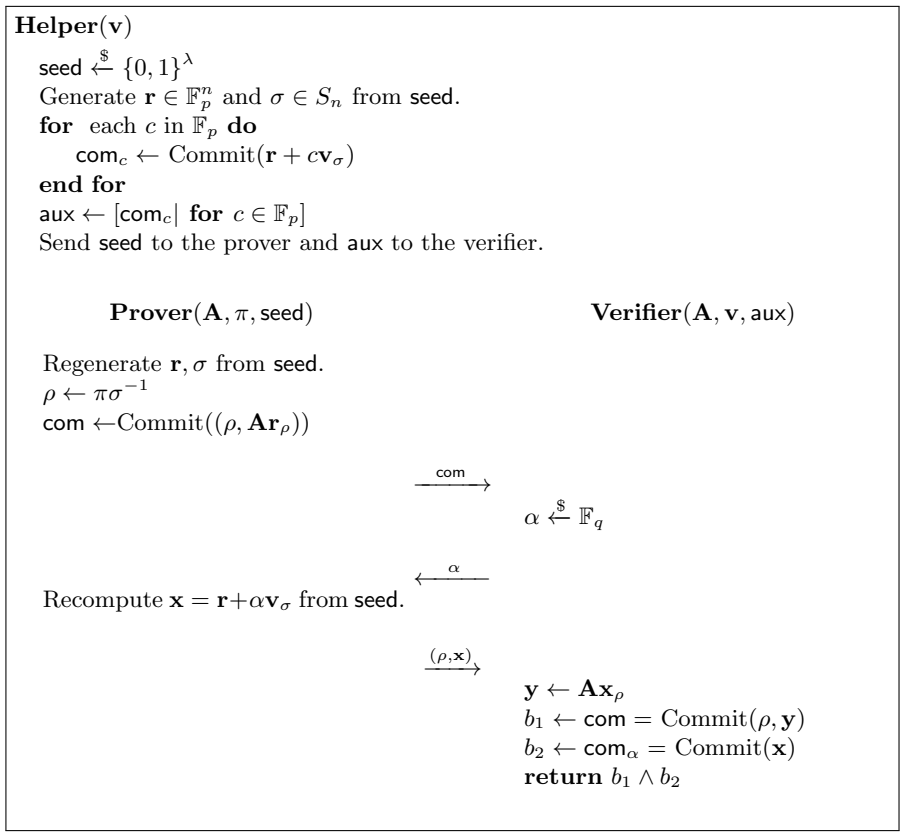

Fig. 4. A sigma protocol with helper for proving knowledge of a solution to the PKP problem.

5.1 Basic transform

The idea is to let the prover pick k seeds $\mathsf{seed}_1, \cdots, \mathsf{seed}_k$ and generate k sets of auxiliary information $\mathsf{aux}_i = \text{Setup}(\mathsf{seed}_i)$ which the prover sends to the verifier, along with the first messages of the protocol $\mathsf{com}_i = P_1(x, w, \mathsf{seed}_i)$ for all i from 1 to k. The verifier then picks a random index I and a single challenge $ch \in \mathcal{C}$ and sends this to the prover. The prover then sends seed_i for $i \neq I$ as well as a response rsp to the challenge at index I. Using the seeds, the verifier then checks if all the auxiliary information $\mathsf{aux}_{i \neq I}$ was generated properly and checks if rsp is a correct response to the challenge at index I. The details of the protocol are displayed in Fig. 5. We prove that this is a honest-verifier zero knowledge protocol with soundness error $\max(\frac{1}{k}, \frac{1}{|\mathcal{C}|})$.

Theorem 3. *Let (Setup, P_1, P_2, V) be a sigma protocol with helper and challenge space $\mathcal{C}$, if the used commitment scheme is hiding, then the protocol of Fig. 5 is an honest-verifier zero knowledge proof of knowledge with challenge space $\{1, \cdots, k\} \times \mathcal{C}$ and $\max(k, |\mathcal{C}|) + 1$-special soundness (and hence it has soundness error $\max(\frac{1}{k}, \frac{1}{|\mathcal{C}|})$).*

Proof. We prove completeness, special soundness and special honest-verifier zero knowledge separately.

Completeness: Follows immediately from the completeness of the underlying Sigma protocol with trusted setup.

$(\max(k, |\mathcal{C}|)+1)$-special Soundness: If there are $\max(k, |\mathcal{C}|)+1$ valid transcripts then there are at least two valid transcripts with different values of I, which implies that all k setups were done honestly. The pigeon hole principle says there are at least two accepted transcripts with the same value of I, but different ch, so the extractor can use special soundness of the underlying Sigma protocol with trusted setup to extract a witness w.

Special Honest-verifier Zero Knowledge: On input (I, ch), the simulator generates all the setups honestly, and commits to random dummy values to create the commitments $\mathsf{com}_{i \neq I}$. The simulator then uses the simulator of the underlying sigma protocol with trusted setup to simulate the transcript at index I. Indistinguishability follows from the hiding property of the commitment scheme and the honest-verifier zero knowledge property of the underlying sigma protocol with trusted setup.

6 Optimizations

In this section we describe optimizations for the MQ and PKP zero-knowledge proofs with trusted setup, as well as for the transformation that removes the trusted setup.

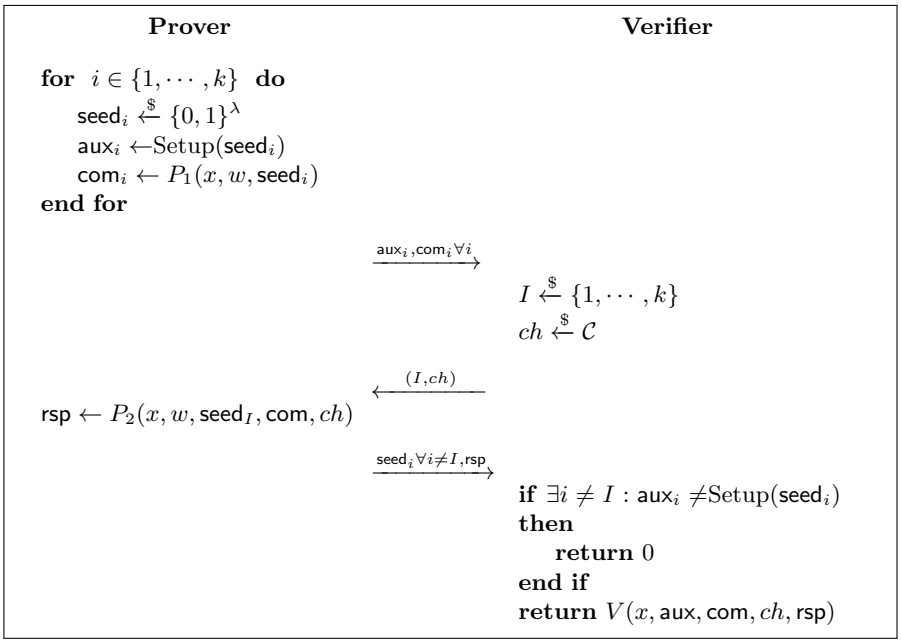

Fig. 5. A zero knowledge proof (without trusted setup) from a Sigma protocol with trusted setup.

Hashing and Merkle trees. In both the MQ proof and the PKP proof the auxiliary information consists of q commitments com_i for $i \in \mathbb{F}_q$, but only one of these commitments, com_α, is opened in each honest execution of the protocol. To reduce the communication cost (and hence the signature size after the Fiat-Shamir transform) we can build a Merkle tree on these commitments and only send the root of the tree. Then the prover includes in his response the $\lceil \log_2(q) \rceil$ nodes of the Merkle tree required to reconstruct the root of the Merkle tree.

When we are doing the transformation to get rid of the trusted party, we do not have to send all the k roots separately. Instead, it suffices to send a hash of all the roots to the verifier. Then during verification the verifier recomputes all the roots (either from seed_i if $i \neq I$, or through the verification algorithm if $i = I$) and hashes the roots to verify that they were correct.

The prover sends k commitments com_i, but only the commitment com_I is used. Therefore, similar to the first optimization, the prover can build a Merkle tree on his commitments and send the root to the verifier. Then, he includes com_I and some nodes of the Merkle tree in his response, so the verifier can recompute the root and authenticate com_I.

Sending less seeds. The prover chooses k seed values, and sends all but one of these seeds to the verifier. We can use a tree strategy to reduce the

communication cost. The prover constructs a binary tree of seed values. First, he picks the value of the root at random. Then, the value of each internal node is used to seed a PRNG which generates the values of its two children. In the end, the leaf nodes act as the seed_i values. Now, instead of sending $k-1$ seed values, the prover can send $\lceil \log_2(k) \rceil$ node values in the tree and the prover can recompute the $k-1$ seeds himself (but not seed_I).

Smaller challenge space. For some applications the finite field $\mathbb{F}_q$ is so large that it would not be practical to compute Merkle trees of size q. In that case we can simply reduce the challenge space to some subset of $\mathbb{F}_q$ of size $q' \leq q$. The soundness error of the scheme then becomes $1/q'$ instead of $1/q$.

Beating parallel repetition. The basic scheme has soundness error $\frac{1}{q'}$, so to reach a soundness error of $2^{-\lambda}$ we would need to perform $r = \left\lceil \frac{\lambda}{log_2(q')} \right\rceil$ parallel executions of the protocol. The optimization of Katz et al. [13] gives a more efficient approach: The idea is that instead of letting the verifier choose 1 out of k setups to execute, we now let him choose τ out of M setups to execute. Now suppose a cheating prover does $e \leq \tau$ out of the M setups incorrectly. Since he cannot produce seed_i values for the cheated setups, he can only convince the verifier if all the setups in which he cheated end up being executed. This happens with probability $\binom{M-e}{\tau-e} \cdot \binom{M}{\tau}^{-1}$. Then, the prover still needs to generate responses for $\tau - e$ honest setups, which he can do with probability at most $\left(\frac{1}{q'}\right)^{\tau-e}$. Therefore the soundness error of the adapted scheme is bounded by

$$\max_{0 \leq e \leq \tau} \frac{\binom{M-e}{\tau-e}}{\binom{M}{\tau} q'^{\tau-e}}.$$

For a more formal proof we refer to [13].

Example 1. Suppose $q = 128$, then without the optimization we would need 19 parallel executions of the basic protocol to reach a soundness error of 2^{-128}, which amounts to $19 * 128 = 2432$ setups and 19 executions of the protocol. With the optimization it turns out that 916 setups and 20 executions are sufficient. So in this case the optimization reduces the number of setups by a factor 2.6 at the cost of a single extra execution.

7 Signature schemes

In this section we apply the Fiat-Shamir transformation to the zero knowledge proofs for MQ and PKP (after applying the transformation of Sect. 5) to obtain 2 signature schemes. We call these schemes the "MUltivariate quaDratic

FIat-SHamir" scheme (MUDFISH) and the "ShUffled Solution to Homogeneous linear SYstem FIat-SHamir" scheme (SUSHSYFISH). First we observe that the recent results on Post-Quantum Fiat-Shamir by Don et al. [6] apply and thus that our signature scheme are provably secure in the QROM (with non-tight reductions). We then give some generic optimizations for the signature scheme and parameter choices for MUDFISH and SUSHSYFISH. We provide a proof-of-concept implementation to show that MUDFISH and SUSHSYFISH are more efficient than existing signature schemes based on the MQ and PKP assumptions (i.e. MQDSS and PKP-DSS respectively) in terms of signature size and speed (on the NIST reference platform).

7.1 Fiat-Shamir transform

The following follows immediately from [6].

Theorem 4. *Assume that a hash function modelled as a Quantum Random Oracle is used as commitment scheme and to derive the challenges, then the MUDFISH and SUSHSYFISH signature schemes are strongly existentially unforgeable in the QROM.*

Proof. The argument is the same as for the fish scheme (the FS variant of Picnic), see Sect. 6.1 of [6].

7.2 Optimizing the signature scheme

Better commitments. Since all the committed values have sufficiently large min-entropy we can use a direct evaluation of a hash function as a commitment scheme. This reduces the communication cost because we don't need randomness for decommitment. Implementing commitments in this way we no longer have zero-knowledge, but for the application of signatures this suffices. Katz et al [13]. observe that if commitments are implemented in this way and the commitment hash function is modelled as a classical random oracle, then a witness can be extracted from a forgery without using the forking lemma. Thus, we can have a tighter security proof in this case.

Slow hash function. Because the QROM security proofs are very non-tight it would not be practical to choose parameters in such a way that security is guaranteed by the proofs. Instead, as is customary, we assume that the probability of a successful attack is at most $H \times E$, where H is the number of hash function evaluations that an attacker makes, and E is the soundness error of the zero knowledge proof. So usually one would choose the parameters such that $E \leq 2^{-\lambda}$. In our implementation we choose a hash function that is a factor 2^k slower than a standard hash function (e.g. SHA-3), therefore it suffices to take our parameters such that $E \leq 2^{-\lambda+k}$. We pick k in such a way that the time spent evaluating the slow hash function is small compared to the total signing and verification time. Since we can take smaller parameters this optimization slightly reduces both the signature size and the signing and verification time.

7.3 MUDFISH

Parameter choices For ease of implementation, we have chosen to use the same finite field $\mathbb{F}_4$ for all the parameter sets. We use MQ systems with the same number of variables as equations, because those are the hardest to solve. For each NIST security level, we pick $n = m$ as the smallest multiple of 4 (for an easier implementation) such that the task of solving a random MQ system has the required hardness. To determine the hardness of the MQ instance we consider 2 solving methods. The Crossbred algorithm of Joux and Vitse [12], which is more suitable to solve MQ instances over small finite fields, and the HybridF5 algorithm, which is more suitable for larger finite fields. The crossbred algorithm (with $d = 1$) starts with a preprocessing phase in which one computes at least k equations which are linear in the first k variables (but of higher degree in the remaining variables), where k is a parameter of the algorithm. Then, in the exhaustive search phase, one iterates over all the q^{n-k} possible values for the remaining $n - k$ variables and solves a k-by-k linear system to get the values of the first k variables. The exhaustive search phase dominates the execution time, and hence we can estimate the time complexity of this approach by

$$q^{n-k_{\max}} k_{\max}^3 ,$$

where $k_{\max}$ is the largest value of k, for which it is possible to do the preprocessing phase. To estimate the complexity of HybridF5 we use the same methodology as Beullens et al. [3]. Our parameter choices for the MQ problem are displayed in Table 1.

We still need to pick parameters for the ZK proof (i.e. τ, the number of executions, M, the number of setups and 2^k, the slowdown factor for the hash function used in the Fiat-Shamir transform). We pick k such that the evaluation time of the hash function is small compared to the total signing time (e.g. 2%). The choice of τ and M allows for a trade-off: If one is willing to increase τ, which mainly impacts signature size, then one can decrease M, which mainly impacts signing and verification time.

NIST PQC Security Level	q	$n = m$	Crossbred ($d = 1$) $k_{\max}$	D	complexity	Hybrid F5 complexity
I	4	84	16	10	2^{146}	2^{157}
III	4	116	20	13	2^{207}	2^{218}
V	4	152	23	14	2^{272}	2^{282}

Table 1. parameters for the MQ problem used by MUDFISH, and the complexity of solving them with the Crossbred algorithm and the Hybrid F5 algorithm.

NIST PQC Security Level	q	n	τ	M	k	$\lvert\mathsf{pk}\rvert$ (B)	$\lvert\mathsf{sk}\rvert$ (B)	$\lvert\mathsf{sig}\rvert$ (KB)	KeyGen (Mc)	Sign (Mc)	Verify (Mc)
I	4	84	64	163	8	37	16	11.2	2.1	13.5	11.1
III	4	116	92	332	9	53	24	25.7	5.4	39.3	37.5
V	4	152	128	384	10	70	32	45.5	12.3	116	97.5

Table 2. parameters for MUDFISH, key and signature sizes and performance measurements (average over 1000 signatures).

Implementation results The signing and verification algorithms require to do a lot of setups and executions of the ZK proof on independent data. We take advantage of this by fitting data from 64 independent rounds into one word. Hence, we can do 64 setups or 64 executions of the protocol in parallel on a 64-bit machine. Our proof-of-concept implementation uses SHAKE256 as hash function and to expand randomness. The performance results of the implementation are displayed in Table 2. We see that MUDFISH is more efficient than MQDSS: Comparing the parameter sets that achieve NIST security level I, the signatures of MUDFISH are 46% smaller than those of MQDSS. At the same time, the signing and verification speed of our proof-of-concept implementation is 49% and 44% faster than those of the optimized implementation submitted to the second round of the NIST PQC standardization project. We leave the task of making an AVX2 optimized implementation of MUDFISH and comparing its performance to the AVX2 optimized implementation of MQDSS for future work.

7.4 SUSHSYFISH

Parameter choices An advantage of building cryptography on PKP is that the best attack algorithms are quite simple and easy to analyze. We use the PKP parameter sets proposed by Faugère et al. [7] to achieve the NIST security levels 1,3 and 5. The choice of the remaining parameters q', τ and M allows for a trade-off between signature size and signing and verification speed. For each of the NIST PQC security levels 1,3 and 5 we propose a parameter set which aims to be fast ($q' = 4$), a parameter sets which aims to have small signatures $q' = 128$ and an intermediate parameter set $q' = 16$.

NIST PQC Security level		q	n	m	q'	τ	M	k	$\lvert\mathsf{pk}\rvert$ (B)	$\lvert\mathsf{sk}\rvert$ (B)	$\lvert\mathsf{sig}\rvert$ (KB)	KeyGen (Mc)	Sign (Mc)	Verify (Mc)
I	Fast	997	61	28	4	61	204	8	72	16	14.8	0.1	6.0	4.2
	Middle	997	61	28	16	33	242	9	72	16	11.6	0.1	15.4	12.7
	Compact	997	61	28	128	16	1011	16	72	16	10.0	0.1	468	457
III	Fast	1409	87	42	4	93	310	9	108	24	35.0	0.2	13.2	9.3
	Middle	1409	87	42	16	49	406	10	108	24	27.0	0.2	37.8	31.5
	Compact	1409	87	42	128	28	1024	17	108	24	24.0	0.2	740	717
V	Fast	1889	111	55	4	128	384	10	142	32	61.3	0.4	17.9	12.3
	Middle	1889	111	55	16	64	607	12	142	32	47.3	0.4	63.5	53.5
	Compact	1889	111	55	128	36	2048	18	142	32	41.5	0.4	1536	1503

Table 3. parameters for SUSHSYFISH, key and signature sizes and performance measurements (average over 1000 signatures).

Implementation results Our implementation uses SHAKE256 as hash function and to expand randomness. The signing and verification times are dominated by the use of SHAKE256, and hence there is a lot of potential for speedups by choosing different symmetric primitives or by parallelizing independent calls of the SHAKE function. The key and signature sizes and the performance measurements for the 9 proposed parameter sets are displayed in Table 3. We see that SUSHSYFISH is more efficient than PKP-DSS. For NIST PQC security level I, the signatures of the "Middle" parameter set are 30% smaller, while both implementations have roughly the same speed (SUSHYFISH is 14% and 13% faster for signing and verification respectively).

8 Zero Knowledge proofs for lattice statements

The decomposition-extension technique of ling et al. [14] embeds the Inhomogeneous Short Integer Solution (ISIS) and ring-ISIS problems into the inhomogeneous PKP problem (i.e. given $\mathbf{A}, \mathbf{v}$ and $\mathbf{t}$, find a permutation π such that $\mathbf{A}\mathbf{v}_\pi = \mathbf{t}$, for some target $\mathbf{t}$ that is not necessarily zero). This can be used to turn a Zero Knowledge proof for inhomogeneous PKP into a zero knowledge proof for ISIS and ring-ISIS. In this section we use our zero knowledge proof for PKP (which is trivially adapted to work for inhomogeneous PKP) to improve the efficiency of Stern-like protocols for ISIS and ring-ISIS.

8.1 Embedding (inhomogeneous) SIS into (inhomogeneous) PKP.

To illustrate the embedding first suppose that $(\mathbf{A}, \mathbf{t}) \in \mathbb{F}_q^{m\times n} \times \mathbb{F}_q^m$ is an instance of the ISIS problem where a solution is a vector $\mathbf{s} \in \mathbb{F}_q^n$ such that $\mathbf{A}\mathbf{s} = \mathbf{t}$ and the coefficients of $\mathbf{s}$ are 0 or 1. In that case we define the extended matrix

$\mathbf{A}' = \begin{pmatrix}\mathbf{A} & \mathbf{0}_{m\times n}\end{pmatrix}$ and a vector $\mathbf{v} \in \mathbb{F}_q$ whose first n entries are 1 and whose last n entries are equal to 0. Then finding a solution to the ISIS instance $(\mathbf{A}, \mathbf{t})$ is equivalent to finding a solution to the inhomogeneous PKP instance $(\mathbf{A}', \mathbf{v}, \mathbf{t})$: Given a solution $\mathbf{s}$ to the ISIS instance it is trivial to find a permutation π such that the first half of $\mathbf{v}_\pi$ equals $\mathbf{s}$, which is then a solution to the inhomogeneous PKP instance. Conversely, if π is a solution to the inhomogeneous PKP instance, then the first half of $\mathbf{v}_\pi$ is a solution to the SIS instance. Therefore, proving knowledge of π is equivalent to proving knowledge of $\mathbf{s}$.

More generally, if the coefficients of $\mathbf{s}$ are required to lie in a range of size B, one can use the decomposition-extension technique [14] to transform an instance of the ISIS problem into an equivalent instance of the inhomogeneous PKP with a matrix $\mathbf{A}'$ with $2n\lceil log_2 B\rceil$ columns [14].

8.2 Generalizing the zero knowledge proof for PKP

It is trivial to generalize our Sigma protocol with helper for PKP to inhomogeneous PKP. Suppose that given $\mathbf{A}, \mathbf{v}$ and $\mathbf{t}$ we want to prove knowledge of a permutation π such that $\mathbf{A}\mathbf{v}_\pi = \mathbf{t}$. Then, the only modification to the Sigma protocol with helper of Fig. 4 we need to make is that the verifier now computes $\mathbf{y}$ as $\mathbf{A}\mathbf{x}_\rho - \alpha\mathbf{t}$ instead of just $\mathbf{A}\mathbf{x}_\rho$. Completeness follows from the fact that

$$\mathbf{A}\mathbf{x}_\rho - \alpha\mathbf{t} = \mathbf{A}(\mathbf{r}_\rho + \alpha\mathbf{v}_\pi) - \alpha\mathbf{t} = \mathbf{A}\mathbf{r}_\rho\,.$$

Special soundness and special honest-verifier zero knowledge are not affected by the modification.

It is also trivial to generalize the protocol to prove knowledge of a solution π of a (inhomogeneous) PKP instance with the additional constraint that π lies in a subgroup $H \subset S_n$. The only modification required is that the prover now samples σ from H instead of from S_n and that the verifier checks that ρ lies in H.

If we use the PKP proof in order to prove knowledge of a solution to the SIS problem as described in the previous subsection, then we can restrict to the subgroup of $S_{2n\lceil log_2 B\rceil}$ generated by the transpositions $(i \quad i + n\lceil log_2 B\rceil)$ for $0 \leq i < n\lceil log_2 B\rceil$. This approach reduces the proof size because elements of the subgroup can be represented with only $n\lceil log_2 B\rceil$ bits, rather than the $log_2((2n\lceil log_2 B\rceil)!)$ bits required to represent an arbitrary permutation.

8.3 Concrete example.

We use the same example as used by del Pino et al. [16], where we prove knowledge of a message that encrypts to a particular Ring-LWE ciphertext. Suppose

we work over the ring $\mathcal{R}_q = \frac{\mathbb{Z}_q[x]}{\langle x^{1024}+1\rangle}$ with q some 13-bit prime. To encrypt the message $\mathbf{m} \in \mathcal{R}_q$ with coefficients in $\{0,1\}$ under the public key $(\mathbf{a}, \mathbf{t})$ one samples small ring elements $\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2$ and computes the ciphertext $(\mathbf{u}, \mathbf{v})$ as follows:

$$\begin{pmatrix} p\mathbf{a} & p & 0 & 0 \\ p\mathbf{t} & 0 & p & 1 \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

Therefore, to prove knowledge of a message that encrypts to $(\mathbf{u}, \mathbf{v})$ it suffices to prove knowledge of a vector in $\mathcal{R}_q^4$ that satisfies the matrix relation, and where the coefficients of the first 3 entries lie in $[-3, 3]$ and the coefficients of the last entry lie in $[0, 1]$.

We find that an embedding of this ISIS problem into inhomogeneous PKP produces a matrix $\mathbf{A}'$ with $n = 2 * (3 + 3 + 3 + 1) * 1024 = 20480$ columns. We choose parameters $\tau = 12, M = 3871, k = 17$ to achieve a soundness error less than 2^{-128}. The proof size is dominated by the vectors and permutations in the proof, of which there is one per execution. Therefore the proof size is roughly equal to

$$\tau(n \log_2(q) + n/2) \text{ bits } ,$$

which amounts to 405 KB. In comparison, we estimate that an application of Stern's protocol would produce proofs of 6.9MB ($\times 17$ larger). On the other hand, del Pino et al. show a method to do this with a proofs size of only 1.25 KB, but their method is not Post-Quantum secure and requires a lot of exponentiations in a cryptographic cyclic group [16].

Acknowledgements This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the European Commission through the Horizon 2020 research and innovation programme under grant agreement H2020-DS-LEIT-2017-780108 FENTEC, by the Flemish Government through FWO SBO project SNIPPET S007619N and by the IF/C1 on Cryptanalysis of post-quantum cryptography. Ward Beullens is funded by an FWO fellowship.

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. Springer (1992)
2. Beullens, W.: FISH. https://github.com/WardBeullens/FISH (2019)
3. Beullens, W., Preneel, B.: Field lifting for smaller UOV public keys. In: International Conference on Cryptology in India. pp. 227–246. Springer (2017)

4. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., 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. ACM (2017)
5. Chen, M.S., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: From 5-pass MQ-based identification to MQ-based signatures. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – Asiacrypt 2016. Lecture Notes in Computer Science, vol. 10032, pp. 135–165. Springer-Verlag Berlin Heidelberg (2016), https://eprint.iacr.org/2016/708
6. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the Quantum Random-Oracle Model. Cryptology ePrint Archive, Report 2019/190 (2019), https://eprint.iacr.org/2019/190
7. Faugère, J.C., Koussa, E., Macario-Rat, G., Patarin, J., Perret, L.: PKP-based signature scheme. Cryptology ePrint Archive, Report 2018/714 (2018), https://eprint.iacr.org/2018/714
8. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Conference on the Theory and Application of Cryptographic Techniques. pp. 186–194. Springer (1986)
9. Georgiades, J.: Some remarks on the security of the identification scheme based on permuted kernels. Journal of Cryptology 5(2), 133–137 (1992)
10. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing. pp. 21–30. ACM (2007)
11. Jaulmes, É., Joux, A.: Cryptanalysis of PKP: a new approach. In: International Workshop on Public Key Cryptography. pp. 165–172. Springer (2001)
12. Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. In: International Conference on Number-Theoretic Methods in Cryptology. pp. 3–21. Springer (2017)
13. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 525–537. ACM (2018)
14. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: International Workshop on Public Key Cryptography. pp. 107–124. Springer (2013)
15. Patarin, J., Chauvaud, P.: Improved algorithms for the permuted kernel problem. In: Annual International Cryptology Conference. pp. 391–402. Springer (1993)
16. del Pino, R., Lyubashevsky, V., Seiler, G.: Short discrete log proofs for FHE and Ring-LWE ciphertexts. In: Public Key Cryptography. pp. 344–373. Springer (2019)
17. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: Annual Cryptology Conference. pp. 706–723. Springer (2011)
18. Shamir, A.: An efficient identification scheme based on permuted kernels. In: Conference on the Theory and Application of Cryptology. pp. 606–609. Springer (1989)
19. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 755–784. Springer (2015)