PROV: PRovable unbalanced Oil and Vinegar
Specification v1.0 – 06/01/2023

Principal submitter:

Louis Goubin
University of Versailles-St-Quentin-en-Yvelines
LMV Laboratory
45 avenue des Etats-Unis
FR-78035 Versailles Cedex
France
Louis.Goubin@uvsq.fr
phone: +33 1 39 25 43 29

Auxiliary submitters:

Benoît Cogliati
Jean-Charles Faugère
Pierre-Alain Fouque
Robin Larrieu
Gilles Macario-Rat
Brice Minaud
Jacques Patarin

Inventors of the cryptosystem: The submitters
Relevant prior work is credited below where appropriate

Owner of the cryptosystem: Same as the submitters

Signature: See independent cover sheet.

Website: prov-sign.github.io

Contents

1 Introduction

PROV is a multivariate cryptography-based signature scheme, and its name stands for PRovable unbalanced Oil-and-Vinegar. As many attacks on Multivariate Cryptography have been published, the confidence in this alternative has been undermined. Consequently, we think it is highly important to support such schemes with a security proof. Since the introduction of UOV, some security proofs appeared at PQCrypto 2011 by Sakumoto *et al*. [SSH11], and more recently by Kosuge and Xagawa [KX22], who also provide a proof in the QROM. Here, we use another proof, which is reminiscent of the security proof of the MAYO signature scheme due to Beullens [Beu22]. The idea is to have a larger oil space than the output of the scheme. This variant is sometimes called UOV$^-$: it corresponds to the UOV scheme where some public equations have been removed. This classical transformation is known as the "Minus" method [**?**] in the literature on Multivariate Cryptography.

1.1 The UOV signature scheme

The Unbalanced Oil-and-Vinegar has been proposed by Kipnis, Patarin and Goubin in [KPG99] about 25 years ago. It is a hash-and-sign signature scheme that follows the GPV framework [GPV08] and its adaptation to multivariate cryptography in [KX22]. From a high level, the UOV algorithm works with two sets of variables: o oil and v vinegar variables. The secret key consists in a tuple of o random quadratic forms $\mathcal{Q}$ that does not involve any product between oil variables. This structure is hidden using a secret linear map $\mathbf{T}$ that mixes oil and vinegar variables. The public key $\mathcal{P}$ is then the composition $\mathcal{Q} \circ T$. The signing procedure of a message msg is very simple: once the vinegar variables are randomly fixed to some vector $\mathbf{v}$, the system $\mathcal{Q}(\mathbf{v}, \mathbf{o}) = \mathsf{hash}(\mathsf{msg})$ becomes linear, and thus easy to solve. In the case where the linear system has no solution, the algorithm simply restarts from the beginning.

1.2 Multivariate Cryptography (MQ)

The main advantage of MQ is to propose very short signature, despite the public key size is quite large, compared to lattice-based schemes or MPC-in-the-head signatures. In this family, the most secure signature scheme is the UOV scheme, which resists to attacks. However, as many attacks have undermined the confidence in MQ, we propose a security proof for PROV.

1.3 Design Rationale

Our goal in this document is to propose a provably secure variant of UOV. In [SSH11], the authors present an UOV variant, dubbed SaltedUOV, whose EUF-CMA-security can be directly linked to the probability of inverting an UOV public key. The main difference with the standard UOV construction is the use of a salt that is hashed alongside the message. In the case where, during the signing procedure, the linear system of equations does not admit any solution, the salt will be resampled instead of the vinegar variables. Even though this simple change makes the UOV scheme provably secure, it can have significant drawbacks. Indeed, the running time of the signature algorithm is now directly linked to the rank of the linear system of equations implied by the choice of $\mathbf{v}$.

With PROV, our goal is to alleviate this problem. In order to do so, we increase the number of oil variables beyond the number of public quadratic forms in such a way that the rank of the linear system of equations will be full with overwhelming probability[1]. This fact and our choice of parameters ensure that the signature algorithm is unlikely to ever need more than one iteration in order to output a signature.

We can summarize our design rationale as follows:

- **simplicity**: one of the main advantages of the UOV family of signature schemes is its simplicity; the algorithms are easy to describe, understand and implement;
- **provable security**: PROV can be proven secure both in the classical and quantum Random Oracle Model, and our choice of parameters is guided by the bound;
- **signature size**: multivariate cryptography is a good candidate for short signature schemes, and PROV is no exception; it is important to note that we make some concession on the signature size in order to attain provable security;
- **reasonable public key size**: we implement well-known optimizations to reduce the public key size, which is arguably one of the weak points of multivariate cryptography;
- **security beyond unforgeability**: we incorporate a simple design tweak based on the BUFF construction [CDF^{+}21] in order to provide several advanced security guarantees (we refer the reader to Section 3.2 for more information).

1.4 Organization of this document

Section 2 is a complete specification of PROV. Section 3 discusses the security of PROV, and describes our parameter set. Section 2.8 deals with implementation issues and possible optimizations. Finally, Section 4 concludes by presenting advantages and limitations of PROV.

1.5 Known Answer Test values

Known answer test values (KAT) for PROV are provided in the submission package.

2 Specification of PROV (2.B.1)

2.1 Preliminaries and notations

Let $\mathbb{F}$ be the Galois field $\mathsf{GF}(2^8)$ with the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. We denote with bold lower-case letters vectors of elements of $\mathbb{F}$. Unless otherwise stated, these will be column vectors. Matrices will be denoted with bold upper-case letters. For any vector $\mathbf{v}$ or matrix $\mathbf{M}$, we denote with $\mathbf{v}^\mathsf{T}$ and $\mathbf{M}^\mathsf{T}$ their respective transpose.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets, and let F be a function from $\mathcal{X}$ to $\mathcal{Y}$. We denote by $\mathsf{Img}(F)$ the image of F, i.e. the set of $y \in \mathcal{Y}$ such that there exists $x \in \mathcal{X}$ satisfying $F(x) = y$. When $\mathcal{X}$ is finite, we denote with $x \leftarrow_{\$} \mathcal{X}$ the sampling of x uniformly at random in $\mathcal{X}$.

[1] We can thus see PROV as a specific instance of SaltedUOV^{-}.

2.2 Parameter space

The main parameters involved in PROV are:

- λ the security parameter of PROV,
- m the number of equations in the public key,
- n the total number of variables,
- δ the difference between the number of oil variables and the number m of equations in the public key,
- $\mathsf{len}_{\mathsf{s_{pk}}}$ the length of the public key seed in bits,
- $\mathsf{len}_{\mathsf{s_{sk}}}$ the length of the private key seed in bits,
- $\mathsf{len}_{\mathsf{salt}}$ the length of salts in bits,
- $\mathcal{H} : \{0,1\}^* \to \mathbb{F}^{\mathsf{m}}$ a hash function,
- $\mathcal{H}' : \{0,1\}^* \to \{0,1\}^{\mathsf{len_{hpk}}}$ a hash function,
- an Extendable-Output Function $\mathcal{XOF}$ that takes as input a bitstring $M \in \{0,1\}^*$ and a non-negative integer d, and outputs a bitstring of length d.

2.3 Key generation

We adopt the description of the UOV algorithm due to Beullens et al. [Beu21, BCH$^+$23]. It relies on an alternative key generation algorithm that compresses public keys without degrading the security of the scheme that was developed in [PBB10, BPB10, PTBW11]. From a high level, the public key consists in a multivariate quadratic map $\mathcal{P} : \mathbb{F}^{\mathsf{n}} \to \mathbb{F}^{\mathsf{m}}$ that is identically zero on a secret $(\mathsf{m} + \delta)$-dimensional vector subspace $O \subset \mathbb{F}^{\mathsf{n}}$:

$$\forall \mathbf{o} \in O, \mathcal{P}(\mathbf{o}) = 0. \tag{1}$$

The key generation algorithm starts with the choice of O under the form of a matrix $\begin{pmatrix} \mathbf{O}^\mathsf{T} & \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix}^\mathsf{T}$ whose columns form a basis of O. The matrix $\mathbf{O} = \mathsf{Expand}_{\mathbf{O}}(\mathsf{s_{sk}}) \in \mathbb{F}^{(\mathsf{n}-\mathsf{m}-\delta)\times(\mathsf{m}+\delta)}$ is generated by deterministically expanding a uniformly random secret key seed $\mathsf{s_{sk}}$ of length $\mathsf{len}_{\mathsf{s_{sk}}}$. The next step consists in the sampling of the public quadratic map $\mathcal{P} = (p_1, \ldots, p_{\mathsf{m}})$. Each quadratic form p_i can be uniquely represented by an upper triangular matrix $\mathbf{P}_i$ such that

$$\forall \mathbf{x} \in \mathbb{F}^{\mathsf{n}}, p_i(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{P}_i\mathbf{x}.$$

Each matrix $\mathbf{P}_i$ will be decomposed into three blocks $\mathbf{P}_i^1 \in \mathbb{F}^{(\mathsf{n}-\mathsf{m}-\delta)\times(\mathsf{n}-\mathsf{m}-\delta)}$, $\mathbf{P}_i^2 \in \mathbb{F}^{(\mathsf{n}-\mathsf{m}-\delta)\times(\mathsf{m}+\delta)}$, and $\mathbf{P}_i^3 \in \mathbb{F}^{(\mathsf{m}+\delta)\times(\mathsf{m}+\delta)}$, such that

$$\mathbf{P}_i = \begin{pmatrix} \mathbf{P}_i^1 & \mathbf{P}_i^2 \\ \mathbf{0} & \mathbf{P}_i^3 \end{pmatrix},$$

where both $\mathbf{P}_i^1$ and $\mathbf{P}_i^3$ are upper triangular. Condition (1) amounts to the fact that the following matrix:

$$\begin{pmatrix} \mathbf{O}^\top & \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{P}_i \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} = \mathbf{O}^\top \mathbf{P}_i^1 \mathbf{O} + \mathbf{O}^\top \mathbf{P}_i^2 + \mathbf{P}_i^3$$

is symmetric, and its diagonal coefficients are 0[2]. Once $\mathbf{P}_i^1$ and $\mathbf{P}_i^2$ are fixed, this condition uniquely determines $\mathbf{P}_i^3$ as this matrix is upper triangular. Hence, we can simply derive the coefficients of $\mathbf{P}_i^1$ and $\mathbf{P}_i^2$ deterministically from a uniformly random public seed $\mathsf{s_{pk}}$ of length $\mathsf{len}_{\mathsf{s_{pk}}}$, and then fix $\mathbf{P}_i^3 = \mathsf{Sym}(-\mathbf{O}^\top \mathbf{P}_i^1 \mathbf{O} - \mathbf{O}^\top \mathbf{P}_i^2)$, where $\mathsf{Sym}(\mathbf{M})$ is the unique upper triangular matrix such that $\mathbf{M} + \mathsf{Sym}(\mathbf{M})$ is symmetric and its diagonal coefficients are 0. The public key will then consist in the tuple $(\mathsf{s_{pk}}, \{\mathbf{P}_i^3\}_{i=1,\ldots,\mathsf{m}})$. The (compressed) private key simply consists in the tuple $(\mathsf{s_{pk}}, \mathsf{s_{sk}}, \mathsf{hpk})$, where hpk denotes the hash of the public key. In the following section, we will also describe an expanded private key that will make the signature computation more efficient. Overall, the size in bits of a PROV public key is

$$\frac{\mathsf{m}(\mathsf{m}+\delta)(\mathsf{m}+\delta+1)}{2}\,(1 + \lfloor \log_2(|\mathbb{F}|) \rfloor) + \mathsf{len}_{\mathsf{s_{pk}}},$$

while the size in bits of a secret key is

$$\mathsf{len}_{\mathsf{s_{sk}}} + \mathsf{len}_{\mathsf{s_{pk}}} + \mathsf{len}_{\mathsf{hpk}}.$$

A pseudocode description of key generation can be found in Algorithm 1.

Algorithm 1 Key generation algorithm.

```
procedure KeyGeneration
    $\mathsf{s_{sk}} \leftarrow_\$ \{0,1\}^{\mathsf{len}_{\mathsf{s_{sk}}}}$
    $\mathsf{s_{pk}} \leftarrow_\$ \{0,1\}^{\mathsf{len}_{\mathsf{s_{pk}}}}$
    $\mathbf{O} \leftarrow \mathsf{Expand}_{\mathbf{O}}(\mathsf{s_{sk}})$
    $(\mathbf{P}_i^1, \mathbf{P}_i^2)_{i=1,\ldots,\mathsf{m}} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{s_{pk}})$
    for $i = 1$ to m do
        $\mathbf{P}_i^3 \leftarrow \mathsf{Sym}(-\mathbf{O}^\top \mathbf{P}_i^1 \mathbf{O} - \mathbf{O}^\top \mathbf{P}_i^2)$
    $\mathsf{pk} \leftarrow (\mathsf{s_{pk}}, (\mathbf{P}_i^3)_{i=1,\ldots,\mathsf{m}})$
    $\mathsf{hpk} \leftarrow \mathcal{H}'(\mathsf{pk})$
    $\mathsf{sk} \leftarrow (\mathsf{s_{pk}}, \mathsf{s_{sk}}, \mathsf{hpk})$
    return $(\mathsf{pk}, \mathsf{sk})$
```

2.4 Signature computation

Let $\mathsf{msg} \in \{0,1\}^*$ be a message to be signed. The goal of the signature procedure is to find a vector $\mathbf{s} \in \mathbb{F}^{\mathsf{n}}$ such that $\mathcal{P}(\mathbf{s}) = \mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt})$, where salt is a bitstring of length $\mathsf{len}_{\mathsf{salt}}$. It will be computed as

$$\mathbf{s} = \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o} \tag{2}$$

[2] Recall that $\mathbb{F}$ is a field of characteristic 2. This would otherwise correspond to the matrix being skew-symmetric.

with $\mathbf{v} \in \mathbb{F}^{\mathsf{n}-\mathsf{m}-\delta}$, and $\mathbf{o} \in \mathbb{F}^{\mathsf{m}+\delta}$. The signature will then consist in the pair $(\mathsf{salt}, \mathbf{s})$, whose length in bits is

$$\mathsf{n}\left(1 + \lfloor \log_2(|\mathbb{F}|) \rfloor\right) + \mathsf{len}_{\mathsf{salt}}.$$

Note that, for $i = 1, \ldots, \mathsf{m}$, one has

$$p_i(\mathbf{s}) = p_i\left(\begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix}\right) + \begin{pmatrix} \mathbf{v}^\mathsf{T} & 0 \end{pmatrix} (\mathbf{P}_i + \mathbf{P}_i^\mathsf{T}) \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o} + p_i\left(\begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o}\right),$$

with $p_i\left(\begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o}\right) = 0$ by construction. Hence, one gets

$$p_i(\mathbf{s}) = \mathbf{v}^\mathsf{T}\mathbf{P}_i^1\mathbf{v} + \mathbf{v}^\mathsf{T}\left(\left(\mathbf{P}_i^1 + \mathbf{P}_i^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_i^2\right)\mathbf{o}.$$

Overall, once $\mathbf{v}$ and salt have been fixed, computing the signature $\mathbf{s}$ of the message m corresponds to solving the following system of linear equations:

$$\begin{cases} \mathbf{v}^\mathsf{T}\left(\left(\mathbf{P}_1^1 + \mathbf{P}_1^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_1^2\right)\mathbf{o} &= h_1 - \mathbf{v}^\mathsf{T}\mathbf{P}_1^1\mathbf{v} \\ & \vdots \\ \mathbf{v}^\mathsf{T}\left(\left(\mathbf{P}_\mathsf{m}^1 + \mathbf{P}_\mathsf{m}^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_\mathsf{m}^2\right)\mathbf{o} &= h_\mathsf{m} - \mathbf{v}^\mathsf{T}\mathbf{P}_\mathsf{m}^1\mathbf{v}, \end{cases} \tag{3}$$

with $(h_1, \ldots, h_\mathsf{m}) = \mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}) \in \mathbb{F}^\mathsf{m}$. This process can be repeated by sampling new salt values until the system (3) admits solutions. Then, a vector $\mathbf{o}$ can be chosen uniformly at random from the set of all solutions of (3). Note that, in order to speed up the computation, it is possible to store the secret matrices $\mathbf{S}_i = \left(\mathbf{P}_i^1 + \mathbf{P}_i^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_i^2$ alongside the secret seed s_{sk}. We refer to $(\mathsf{sk}, (\mathbf{S}_i)_{i=1,\ldots,m})$ as the *expanded* secret key esk, whose size in bits is

$$\mathsf{m}(\mathsf{n}-\mathsf{m}-\delta)(\mathsf{m}+\delta)\left(1 + \lfloor \log_2(|\mathbb{F}|) \rfloor\right) + \mathsf{len}_{\mathsf{hpk}} + \mathsf{len}_{\mathsf{s}_{\mathsf{pk}}} + \mathsf{len}_{\mathsf{s}_{\mathsf{sk}}}.$$

A pseudocode description of this step can be found in Algorithm 2.

Algorithm 2 Signature algorithm.

```
procedure $\textsc{Sign}(\mathsf{sk}, \mathsf{msg})$
    $(\mathsf{s_{pk}}, \mathsf{s_{sk}}, \mathsf{hpk}) \leftarrow \mathsf{sk}$
    $\mathbf{O} \leftarrow \mathsf{Expand}_{\mathbf{O}}(\mathsf{s_{sk}})$
    $(\mathbf{P}_i^1, \mathbf{P}_i^2)_{i=1,\dots,\mathsf{m}} \leftarrow \mathsf{Expand_{pk}}(\mathsf{s_{pk}})$
    $\mathbf{v} \leftarrow_{\$} \mathbb{F}^{\mathsf{n}-\mathsf{m}-\delta}$
    repeat
        $\mathsf{salt} \leftarrow_{\$} \{0,1\}^{\mathsf{len_{salt}}}$
        $(h_1, \dots, h_{\mathsf{m}}) \leftarrow \mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt})$
        for $i = 1$ to $\mathsf{m}$ do
            $t_i \leftarrow h_i - \mathbf{v}^\mathsf{T}\mathbf{P}_i^1\mathbf{v}$
            $\mathbf{a}_i \leftarrow \mathbf{v}^\mathsf{T}\left(\left(\mathbf{P}_i^1 + \mathbf{P}_i^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_i^2\right)$
        $\mathbf{A_v} \leftarrow (\mathbf{a}_i)_{i=1,\dots,\mathsf{m}}$
        $\mathbf{t} \leftarrow (t_i)_{i=1,\dots,\mathsf{m}}$
        $S \leftarrow \mathsf{LinSolve}(\mathbf{A_v}, \mathbf{t})$
    until $S \neq \emptyset$
    $\mathbf{o} \leftarrow_{\$} S$
    $\mathbf{s} \leftarrow \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o}$
    return $(\mathsf{salt}, \mathbf{s})$
```

2.5 Signature verification

The signature verification simply consists in verifying that the equation $\mathcal{P}(\mathbf{s}) = \mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt})$ holds. A pseudocode description of this step can be found in Algorithm 3.

Algorithm 3 Signature verification.

```
procedure $\textsc{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig})$
    $(\mathsf{s_{pk}}, (\mathbf{P}_i^3)_{i=1,\dots,\mathsf{m}}) \leftarrow \mathsf{pk}$
    $(\mathbf{P}_i^1, \mathbf{P}_i^2)_{i=1,\dots,\mathsf{m}} \leftarrow \mathsf{Expand_{pk}}(\mathsf{s_{pk}})$
    $(\mathsf{salt}, \mathbf{s}) \leftarrow \mathsf{sig}$
    $\mathsf{hpk} \leftarrow \mathcal{H}'(\mathsf{pk})$
    $\mathbf{h} \leftarrow \mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt})$
    for $i = 1$ to $\mathsf{m}$ do
        $\mathbf{P}_i \leftarrow \begin{pmatrix} \mathbf{P}_i^1 & \mathbf{P}_i^2 \\ \mathbf{0} & \mathbf{P}_i^3 \end{pmatrix}$
        $t_i \leftarrow \mathbf{s}^\mathsf{T}\mathbf{P}_i\mathbf{s}$
    return $(t_i)_{i=1,\dots,\mathsf{m}} \overset{?}{=} \mathbf{h}$
```

2.6 Determinization

For ease of exposition, the above description presents a variant of PROV where various quantities are sampled uniformly at random. In reality, all randomnss is ultimately derived from the secret seed and the message. In particular, PROV signatures are deterministic: the signature of a given message is always the same. We now explain in detail how the previous construction is made deterministic.

2.6.1 Hash computations

All hash computations use SHAKE256, from now on written $\mathcal{H}$, with a unique one-byte prefix for domain separation.

1. The public seed is derived from the secret seed with the prefix 0: $\mathsf{s_{pk}} = \mathcal{H}(0 \| \mathsf{s_{sk}})$.
2. The bytes of $\mathbf{P}^1_i$ and $\mathbf{P}^2_i$ are derived from the public seed with the prefix 1, as $\mathcal{H}(1 \| \mathsf{s_{pk}})$. The first bytes are used for $\mathbf{P}^1_i$, and the following ones for $\mathbf{P}^2_i$. See Section 2.6.2 below for information about how bytes are inserted in each matrix.
3. The bytes of matrix $\mathbf{O}$ are derived from the secret seed with the prefix 2, as $\mathcal{H}(2 \| \mathsf{s_{sk}})$.
4. The vinegar value $\mathbf{v}$ is as $\mathbf{v} = \mathcal{H}(3 \| \mathsf{msg})$. The same SHAKE256 instance is then squeezed to produce the initial oil value (used in LinSolve, as explained in Section 2.6.3), and finally to produce successive salt values.
5. The hashed message $\mathbf{h}$ is computed as $\mathbf{h} = \mathcal{H}(4 \| \mathsf{hpk} \| \mathsf{msg} \| \mathsf{salt})$.
6. The hashed public key is computed as $\mathsf{hpk} = \mathcal{H}(5 \| \mathsf{s_{pk}} \| (\mathbf{P}^3_i))$.

2.6.2 Byte order in vectors and matrices

Bytes are inserted into vectors in the same order as they are produced by SHAKE256: the first byte output becomes the first coordinate of the vector. For matrices, bytes are inserted starting from coordinates $(0, 0)$, and proceeding along colums. The choice of inserting along columns is justified by the fact that it is convenient in PROV to represent columns as contiguous memory chunks, rather than rows. This is because key generation involves products of the form $\mathbf{O}^\mathsf{T}\mathbf{P}_i$, where the scalar products are between columns of the original matrices; while signature computation involves products of the form $\left(\mathbf{P}^1_\mathsf{m} + \mathbf{P}^{1\,\mathsf{T}}_\mathsf{m}\right)\mathbf{O}$, where the left multiplicand is symmetric.

2.6.3 Determinization of LinSolve

By design of PROV, the linear system computed during the signing process is expected to have many solutions. The solution output by the signing algorithm is chosen uniformly among the solution set. We now explain how this process is made deterministic.

As explained in Section 2.6.1, the message is used to derive deterministically the vinegar value $\mathbf{v}$, as well as an initial value for the vector $\mathbf{o}$. LinSolve take this initial value as input, and will use it as its random coins. In more detail, LinSolve solves the system by computing its row echelon form. Then it uses back substitution to adjust the coefficients of initial oil value so that it is a solution of the system; but it does so by *only* modifying the entries at positions corresponding to the leading coefficients of the row echelon form. Note that those positions are uniquely determined, and so is the corresponding modification of the coefficients. Also observe that if the initial oil value is uniformly random, the output solution is uniformly random among the solution set by linearity. This solution was adopted because it is easy to implement, and does not depend on low-level details of the linear solver.

2.7 Parameter sets

Variant	λ	n	m	δ	|seed|	|salt|	|hpk|	|sig|	|pk|	|sk|	|esk|
PROV-I	128	136	46	8	16	24	32	160	68326	16	203752
PROV-III	192	200	70	8	24	32	48	232	215694	24	666216
PROV-V	256	264	96	8	32	40	64	304	524192	32	1597568

Table 1: Parameter sets and corresponding key and signature sizes for the PROV signature scheme, in bytes.

2.8 Implementation and Performances (2.B.2)

The focus of PROV is on showing that it is possible to realize a provable variant of the classical UOV cryptosystem, at a modest cost in the choice of parameters. Indeed, n is essentially unchanged, while m is increased by roughly 20% for the lowest security level, and 10% for levels 3 and 5, compared to what the analysis from Section 3.4 deems secure when setting $\delta = 0$. This means practical performance is expected to be close a standard UOV cryptosystem.

A number of implementation details were discussed in Section 2.6. In particular, strided memory access patterns can be largely avoided by representing matrix rows contiguously in memory, thanks to the fact that most matrix-matrix multiplications involve transpositions or symmetric matrices on the left-hand side. The heaviest computational cost comes from the numerous matrix multiplications. This can be optimized by taking advantage of the numerous fast implementations of linear algebra available for various platforms, as well as optimized field operations. Due to our focus on provable security, we have not yet explored those options.

3 Security analysis of PROV

3.1 Security reduction (part of 2.B.4)

Security models. In this document, we consider the standard Existential UnForgeability under Chosen-Message Attack (EUF-CMA) notion for signature schemes. In this scenario, the adversary gets a PROV public key, and has access to the corresponding signing oracle that can be queried at most 2^{64} times. Its goal is to generate a valid signature for a new message. Formally, one has the following definition.

Definition 1 (EUF-CMA security)**.** *Let $\mathcal{H}$ be a random oracle, and let $\mathcal{A}$ be an adversary. The advantage of $\mathcal{A}$ against the* EUF-CMA *security of a signature scheme $S = (S.\textsc{KeyGeneration}, S.\textsc{Sign}^{\mathcal{H}}, S.\textsc{Verify}^{\mathcal{H}})$ is defined as*

$$\mathsf{Adv}_S^{\mathsf{EUF\text{-}CMA}}(\mathcal{A}) = \Pr\left[S.\textsc{Verify}(\mathsf{pk}, \mathsf{msg}, \mathsf{sig}) = \top \textit{ and } S.\textsc{Sign}^{\mathcal{H}}(\mathsf{sk}, \cdot) \textit{ was never queried on } \mathsf{msg}\right],$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow S.\textsc{KeyGeneration}()$ *and* $(\mathsf{msg}, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathcal{H}, S.\textsc{Sign}^{\mathcal{H}}(\mathsf{sk}, \cdot)}(\mathsf{pk})$.

In this section, we provide proofs that PROV is EUF-CMA-secure both in the Random Oracle Model (ROM) and the Quantum ROM (QROM) as long as the UOV problem is hard to solve for our choice of parameters. In the quantum setting, we will rely on a generic result from [KX22]. To

```
$\mathsf{PS}_b$
$(\mathrm{F},\mathrm{I}) \leftarrow \textsc{Gen}(1^\lambda)$
$b^* \leftarrow \mathcal{A}^{\mathsf{Sample}_b}(\mathrm{F})$
return $b^*$
```

```
$\mathsf{Sample}_0$
$z_i \leftarrow \mathrm{I}^1()$
repeat
    $y_i \leftarrow_\$ \mathcal{Y}$
    $x_i \leftarrow \mathrm{I}^2(z_i, y_i)$
until $x_i \neq \bot$return $x_i$
```

```
$\mathsf{Sample}_1$
$x_i \leftarrow \textsc{SampDom}(\mathrm{F})$
return $x_i$
```

Figure 1: Preimage sampling game.

this end, it is necessary to introduce the notion of Weak Preimage-Sampleable Function (WPSF), as tailored to the Multivariate Cryptography setting in [KX22].

Definition 2 (Weak Preimage-Sampleable Function (WPSF) [KX22])**.** *A WPSF* T *consists of four algorithms:*

- $\textsc{Gen}$*: this algorithm takes as input a security parameter and outputs a function* $\mathrm{F} : \mathcal{X} \to \mathcal{Y}$ *with a trapdoor* I*;*
- F*: this algorithm takes as input a value* $x \in \mathcal{X}$ *and deterministically outputs* $\mathrm{F}(x)$*;*
- $\mathrm{I} = (\mathrm{I}^1, \mathrm{I}^2)$*: the first algorithm takes no input and samples a value* $z \in \mathcal{Z}$*; the second one algorithm takes as input* $z \in \mathcal{Z}$*,* $y \in \mathcal{Y}$*, and outputs* $x \in \mathcal{X}$ *such that* $\mathrm{F}(x) = y$*, or outputs* $\bot$ *in case of failure;*
- $\textsc{SampDom}$*: this algorithm takes as input a function* $\mathrm{F} : \mathcal{X} \to \mathcal{Y}$ *and outputs* $x \in \mathcal{X}$*.*

The security of a WPSF is defined as follows.

Definition 3 (PS security [KX22])**.** *Let* T *be a WPSF. The advantage of an adversary* $\mathcal{A}$ *against the* PS *security of* T *is defined as follows:*

$$\mathsf{Adv}_T^{\mathsf{PS}}(\mathcal{A}) = \left| \Pr\left[\mathsf{PS}_0^{\mathcal{A}} = 1\right] - \Pr\left[\mathsf{PS}_1^{\mathcal{A}} = 1\right] \right|,$$

where PS_0 *and* PS_1 *are the games defined in Figure 1.*

Finally, we define the INV game against a WPSF T as follows.

Definition 4 (INV security)**.** *Let* $\mathcal{A}$ *be an* INV *adversary against* T*, trying to inverse the public function* F*. We define its advantage as*

$$\mathsf{Adv}_{\mathsf{T}}^{\mathsf{INV}}(\mathcal{A}) = \Pr\left[\mathrm{F}(x) = y\right],$$

with $(\mathrm{F}, \cdot) \leftarrow \textsc{Gen}(1^\lambda)$ *and* $y \leftarrow_\$ \mathcal{Y}$*,* $x \leftarrow \mathcal{A}(\mathrm{F}, y)$*.*

Hardness assumptions. The UOV problem has been well-studied since its introduction in 1999. Over the years, multiple slight variants have been introduced. The mathematical problem that underlies PROV is the well-known $\mathsf{UOV}^{\text{-}}$ problem, and can be defined as follows.

Definition 5 (UOV$^-$ problem). *Let $\mathcal{P}$ the quadratic map associated with a* PROV *public key* pk. *The* UOV$^-$ *problem asks to find $\mathbf{s} \in \mathbb{F}^{\mathsf{n}}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$. More formally, the advantage of an adversary $\mathcal{A}$ against the* INV *security of the* UOV$^-$ *is defined as*

$$\mathsf{Adv}^{\mathsf{INV}}_{\mathsf{UOV}^-}(\mathcal{A}) = \Pr\left[\mathcal{P}(\mathbf{s}) = \mathbf{y}\right],$$

where $(\mathsf{pk}, \mathsf{sk}) \leftarrow \textsc{KeyGeneration}()$, $\mathbf{y} \leftarrow_{\$} \mathbb{F}^{\mathsf{m}}$, *and $\mathcal{P}$ is the quadratic map corresponding to* pk.

Note that the UOV$^-$ problem can also be recast as a an inversion (INV) problem for the following WPSF, dubbed $\mathsf{T}_{\mathsf{PROV}}$:

- the GEN algorithm corresponds to the key generation function;
- F is the evaluation of the public quadratic map;
- SAMPDOM samples a value in $\mathbb{F}^{\mathsf{n}}$ uniformly at random;
- I corresponds to the pair $(\mathrm{I}^1, \mathrm{I}^2)$ described in Algorithm 4.

One clearly has $\mathsf{Adv}^{\mathsf{INV}}_{\mathsf{UOV}^-} = \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{PROV}}}$.

Algorithm 4 Algorithms I^1 and I^2 of the $\mathsf{T}_{\mathsf{PROV}}$ WPSF.

```
procedure $\mathrm{I}^1$
    $\mathbf{v} \leftarrow_{\$} \mathbb{F}^{\mathsf{n}-\mathsf{m}-\delta}$
    return $\mathbf{v}$
procedure $\mathrm{I}^2(\mathsf{sk}, \mathbf{v}, \mathbf{y})$
    $(\mathsf{s_{pk}}, \mathsf{s_{sk}}) \leftarrow \mathsf{sk}$
    $\mathbf{O} \leftarrow \mathsf{Expand}_{\mathbf{O}}(\mathsf{s_{sk}})$
    $(\mathbf{P}^1_i, \mathbf{P}^2_i)_{i=1,\ldots,\mathsf{m}} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{s_{pk}})$
    for $i = 1$ to m do
        $t_i \leftarrow \mathbf{y}_i - \mathbf{v}^{\mathsf{T}}\mathbf{P}^1_i\mathbf{v}$
        $\mathbf{a}_i \leftarrow \mathbf{v}^{\mathsf{T}}\left(\left(\mathbf{P}^1_i + \mathbf{P}^{1\,\mathsf{T}}_i\right)\mathbf{O} + \mathbf{P}^2_i\right)$
    $\mathbf{A}_{\mathbf{v}} \leftarrow (\mathbf{a}_i)_{i=1,\ldots,\mathsf{m}}$
    $\mathbf{t} \leftarrow (t_i)_{i=1,\ldots,\mathsf{m}}$
    $S \leftarrow \mathsf{LinSolve}(\mathbf{A}, \mathbf{t})$
    if $S = \emptyset$ then
        return $\perp$
    $\mathbf{o} \leftarrow_{\$} S$
    $\mathbf{s} \leftarrow \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o}$
    return $\mathbf{s}$
```

Classical security. In this paragraph, we model the hash function $\mathcal{H}$ as a random oracle. One has the following result.

Theorem 1. *Let q_s, q_h and t be three positive integers. Let $\mathcal{A}$ (resp. $\mathcal{A}'$) be an adversary against the* EUF-CMA *security of* PROV *(resp.* INV *security of* $\mathsf{T}_{\mathsf{PROV}}$*) that runs in time at most t, makes at most q_s*

signing queries and q_h random oracle queries. Moreover, let N be the random variable corresponding to the overall number of salts that are sampled during all the queries to the signing oracle. Then, for any constant $C > 0$, one has

$$\mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T}_{\mathsf{PROV}}}(\mathcal{A}') = 0.$$

Moreover, there exists an INV *adversary* $\mathcal{B}$ *against* $\mathsf{T}_{\mathsf{PROV}}$ *derived from* $\mathcal{A}$ *such that*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{PROV}}(\mathcal{A}) \leq \frac{(q_h + q_s + 1)}{1 - \frac{q_s(q_h+q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}}} \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{PROV}}}(\mathcal{B}) + \Pr\left[N > C\right] + \frac{C(q_h + q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}}.$$

In particular, one has

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{PROV}}(\mathcal{A}) \leq \frac{(q_h + q_s + 1)}{1 - \frac{q_s(q_h+q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}}} \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{PROV}}}(\mathcal{B}) + \Pr\left[N > q_s\right] + \frac{q_s(q_h + q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}}.$$

Proof. Our proof is based on [SSH11] and [Beu22]. We state it for the sake of completeness. Let us fix three positive integers q_s, q_h and t. Let $\mathcal{A}$ be an adversary against the EUF-CMA-security of PROV that runs in time at most t, makes at most q_s signing queries and q_h random oracle queries. Let also $\mathcal{A}'$ be an adversary against the INV security of $\mathsf{T}_{\mathsf{PROV}}$ that runs in time at most t and makes at most q_s sampling queries.

Let us start by describing the adversary $\mathcal{B}$ against the INV security of $\mathsf{T}_{\mathsf{PROV}}$. It takes an input a PROV public key pk, that can be expanded to a quadratic map $\mathcal{P}$, and maintains a list L that is initially empty, and a counter i that is initially 0. First, $\mathcal{B}$ will sample $\alpha \in \{1, \ldots, \alpha\}$, and then run $\mathcal{A}$, answering its queries as follows:

- on a random oracle query x, $\mathcal{B}$ will answer $\mathbf{h}$ if $(x, \mathbf{h}) \in L$, and otherwise it increments i, and depending on the value of i two cases can occur:
  - if $i = \alpha$, $\mathcal{B}$ outputs $\mathbf{t}$ and adds $(x, \mathbf{t})$ to L;
  - otherwise it samples and outputs a vector $\mathbf{h}$ uniformly at random in $\mathbb{F}^{\mathsf{m}}$ and adds $(x, \mathbf{h})$ to L;
- on a signature query msg, it increments i and samples a new salt value salt $\leftarrow_{\$} \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$; then two cases can occur:
  - if there exists an entry $(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}, \cdot)$ in L, $\mathcal{B}$ aborts;
  - otherwise, it samples $\mathbf{s} \leftarrow_{\$} \mathbb{F}^{\mathsf{n}}$, computes $\mathbf{h} = \mathcal{P}(\mathbf{s})$, adds the tuple $(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}, \mathbf{h})$ to L and answers $(\mathsf{salt}, \mathbf{s})$ to $\mathcal{A}$.

Eventually, $\mathcal{A}$ outputs some forgery $(\mathsf{salt}, \mathbf{s})$ for some message msg. If $(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}, \mathbf{t}) \in L$, then we get $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$, and $\mathcal{B}$ can simply forward the preimage $\mathbf{s}$. Otherwise, $\mathcal{B}$ fails.

We denote G_0 (resp. G_1) $\mathcal{A}$'s EUF-CMA game against PROV (resp. $\mathcal{B}$'s INV game against $\mathsf{T}_{\mathsf{PROV}}$). Let us introduce the following events:

- Coll is the event where, in G_0, the SIGN algorithm samples a salt value (even a value that will be discarded due to an absence of solutions) that collides with a salt value from a previous signature, or one that appears in the list of random oracle queries;
- Abort is the event where $\mathcal{B}$ aborts in G_1;

- $\mathsf{Forgery}$ is the event where $\mathcal{A}$ successfully outputs a forgery in G_1.

The probability of $\mathcal{B}$ succeeding is greater than the probability that it does not abort, that the $\mathsf{Forgery}$ event occurred, and that the forgery actually corresponds to the random oracle output $\mathbf{t}$. Hence, one has

$$\mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T_{PROV}}}(\mathcal{B}) = \Pr\left[\mathsf{G}_1() = 1\right] \geq \frac{1}{q_h + q_s + 1} \Pr\left[\neg\mathsf{Abort}\right] \Pr\left[\mathsf{Forgery}|\neg\mathsf{Abort}\right],$$

i.e.

$$\Pr\left[\mathsf{Forgery}|\neg\mathsf{Abort}\right] \leq \frac{(q_h + q_s + 1)}{1 - \frac{q_s(q_h+q_s)}{2^{\mathsf{len_{salt}}}}} \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T_{PROV}}}(\mathcal{B}). \tag{4}$$

Let also N be the random variable that corresponds to the number of salt samplings in all signature queries. Then, for any constant C, one also has

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{PROV}}(\mathcal{A}) = \Pr\left[\mathsf{G}_0() = 1\right] &\leq \Pr\left[\mathsf{G}_0() = 1|\neg\mathsf{Coll}\right] + \Pr\left[\mathsf{Coll}\right] \\ &\leq \Pr\left[\mathsf{G}_0() = 1|\neg\mathsf{Coll}\right] + \Pr\left[N > C\right] + \Pr\left[\mathsf{Coll}|N \leq C\right] \\ &\leq \Pr\left[\mathsf{G}_0() = 1|\neg\mathsf{Coll}\right] + \Pr\left[N > C\right] + \frac{C(q_h + q_s)}{2^{\mathsf{len_{salt}}}}. \end{aligned} \tag{5}$$

We now argue that $\Pr\left[\mathsf{G}_0() = 1|\neg\mathsf{Coll}\right] = \Pr\left[\mathsf{Forgery}|\neg\mathsf{Abort}\right]$. In order to do so, let us compare the probability distributions of the tuple $(\mathsf{salt}, \mathbf{s}, \mathbf{h})$ that stems from a signature query msg in both games. In particular, this means that $\mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}) = \mathbf{h} = \mathcal{P}(\mathbf{s})$.

Let us denote by S the set of all salt' values in the outputs of previous signature queries, and all salt' values such that there exists a message msg' satisfying $(\mathsf{hpk}||\mathsf{msg}'||\mathsf{salt}', *) \in L$. Since we conditioned on Abort and Coll not occurring, in both games, salt values are sampled uniformly at random in $\{0,1\}^{\mathsf{len_{salt}}} \setminus S$. In game G_1, the probability p_1 of getting this tuple is exactly

$$p_1 = \frac{1}{2^{\mathsf{len_{salt}}} - |S|} \cdot \frac{1}{|\mathbb{F}|^{\mathsf{n}}}.$$

Let us now consider the probability p_0 of sampling the same tuple in game G_0. Recall that $\mathbf{s}$ can be uniquely written as

$$\mathbf{s} = \begin{pmatrix} \mathbf{v} \\ 0 \end{pmatrix} + \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \mathbf{o},$$

with $\mathbf{v} \in \mathbb{F}^{\mathsf{n}-\mathsf{m}-\delta}$ and $\mathbf{o} \in \mathbb{F}^{\mathsf{m}+\delta}$. In Game G_0, the $\textsc{Sign}$ algorithm first has to pick a vinegar value that matches $\mathbf{v}$. This fixes the linear map $\mathbf{A_v}$ that has to be inverted in order to compute the signature. Let $d \leq m$ be the rank of $\mathbf{A_v}$. Since we conditioned on the Coll event not occurring, salts will be sampled uniformly at random outside of S. In particular, this means that $\mathcal{H}$ is never queried on the same input twice. If we denote with salt_j the salt value sampled at the j-th iteration of the loop, the event "the signature algorithm outputs $(\mathsf{salt}, \mathbf{s})$ at the i-th iteration", whose probability will be denoted $p_{0,i}$, corresponds to the following events:

- for iterations $i = 1, \ldots, i-1$, one has $\mathcal{H}(\mathsf{msg}||\mathsf{salt}_j) - (\mathbf{v}^{\mathsf{T}}\mathbf{P}^1_i\mathbf{v})_{i=1,\ldots,\mathsf{m}} \notin \mathsf{Img}(\mathbf{A_v})$, which happens with probability $\left(1 - \frac{1}{|\mathbb{F}|^{\mathsf{m}-d}}\right)^{i-1}$;
- $\mathsf{salt}_i = \mathsf{salt}$, which happens with probability $\frac{1}{2^{\mathsf{len_{salt}}}-|S|}$;

- the value $\mathbf{v}$ is sampled for the vinegar variables, which happens with probability $\frac{1}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}}$;
- $\mathcal{H}(\mathsf{hpk}||\mathsf{msg}||\mathsf{salt}_j) = \mathbf{h}$, which happens with probability $\frac{1}{|\mathbb{F}|^{\mathsf{m}}}$;
- the vector $\mathbf{o}$ is sampled among the $|\mathbb{F}|^{\mathsf{m}+\delta-d}$ solutions of the system $\mathbf{A_v o} = \mathbf{h} - (\mathbf{v}^\mathsf{T}\mathbf{P}_k^1\mathbf{v})_{k=1,\ldots,\mathsf{m}}$, which happens with probability $\frac{1}{|\mathbb{F}|^{\mathsf{m}+\delta-d}}$.

Thus, the probability that the signature algorithm outputs $(\mathsf{salt}, \mathbf{s})$ at the i-th iteration of the loop is exactly

$$p_{0,i} = \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}} - |S|} \times \left(1 - \frac{1}{|\mathbb{F}|^{\mathsf{m}-d}}\right)^{i-1} \times \frac{1}{|\mathbb{F}|^{\mathsf{n}+\mathsf{m}-d}}.$$

Summing over all possible values of i yields

$$p_0 = \frac{1}{2^{\mathsf{len}_{\mathsf{salt}}} - |S|} \times \frac{1}{|\mathbb{F}|^{\mathsf{n}}} = p_1.$$

Moreover, since no salts collide, random oracle queries always have different outputs and $\mathcal{B}$ simulates the random oracle perfectly. Hence, the view of $\mathcal{A}$ in both games is identically distributed and

$$\Pr[\mathsf{G}_1() = 0 | \neg\mathsf{Coll}] = \Pr[\mathsf{Forgery} | \neg\mathsf{Abort}].$$

Combining this equality with Eqs (4) and (5) yields the second part of the result. Moreover, we have also proven that both views in the PS game for $\mathsf{T}_{\mathsf{PROV}}$ are identically distributed, which proves the first part of the result. □

Remark 1. PROV *is set up so that the linear system of equations* $\mathbf{A_v o} = \mathbf{t}$ *that has to be solved in the signature computations (Algorithm 2, line 14) has* $\mathsf{m} + \delta$ *variables and* m *equations. This choice has been made in order to significantly reduce the probability that the rank of the system is smaller than* m*. Indeed, as long as the system has full rank,* SIGN *will always need exactly one iteration of the loop. This implies that*

$$\Pr[N > q_s] \le q_s p_{\mathsf{rank}},$$

where p_{rank} *is an upper bound on the probability that the rank of the matrix* $\mathbf{A_v}$ *is at most* $\mathsf{m} - 1$*. Following [SSH11], we will assume that, as long as* $\mathbf{v} \neq \mathbf{0}$*, the rank of the system follows the same distribution as the rank of a uniformly random* $(\mathsf{m} + \delta) \times \mathsf{m}$ *matrix. Intuitively, the i-th row* $\mathbf{a}_i$ *of* $\mathbf{A_v}$ *is computed as*

$$\mathbf{v}^\mathsf{T}\left(\left(\mathbf{P}_i^1 + \mathbf{P}_i^{1\,\mathsf{T}}\right)\mathbf{O} + \mathbf{P}_i^2\right).$$

Hence, since $\mathbf{v}$ *is non-zero and* $\mathbf{P}_i^2$ *is a uniformly random matrix, then so is* $\mathbf{A}$*. This argument has also been used by Beullens in [Beu22]. According to [Lev05], the probability that a random* $\mu \times \nu$ *matrix on a field with cardinality* q *has the rank* $i > 0$ *is equal to*

$$\frac{\prod_{j=\mu-i+1}^{\mu}(1-q^{-j})\prod_{j=\nu-i+1}^{\nu}(1-q^{-j})}{\prod_{j=1}^{i}(1-q^{-j})},$$

where $\mu \le \nu$*. In particular, taking* $\mu = \mathsf{m}$*,* $\nu = \mathsf{m} + \delta$*,* $q = |\mathbb{F}|$ *and* $i = \mathsf{m}$*, one has*

$$p_{\mathsf{rank}} \le \frac{1}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + 1 - \prod_{j=1}^{\mathsf{m}}\left(1 - \frac{1}{|\mathbb{F}|^{\delta+j}}\right) \le \frac{1}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + \frac{1}{|\mathbb{F}|^{\delta}(|\mathbb{F}|-1)}.$$

Corollary 1. *Let q_s, q_h and t be three positive integers. Let $\mathcal{A}$ (resp. $\mathcal{A}'$) be an adversary against the* EUF-CMA *security of* PROV *(resp. the* INV *security of* $\mathsf{T}_{\mathsf{PROV}}$*) that runs in time at most t, makes at most q_s signing queries and q_h random oracle queries. Under the approximation discussed in Remark 1, one has*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{PROV}}(\mathcal{A}) \le \frac{(q_h+q_s+1)}{1-\frac{q_s(q_h+q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}}}\mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{PROV}}}(\mathcal{B}) + \frac{q_s}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + \frac{q_s}{|\mathbb{F}|^{\delta}(|\mathbb{F}|-1)} + \frac{q_s(q_h+q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}},$$

where $\mathcal{B}$ is an INV *adversary derived from $\mathcal{A}$ against* $\mathsf{T}_{\mathsf{PROV}}$.

Post-quantum security. In this paragraph, we consider that the adversary has quantum access to the underlying hash function, and classical access to its signing oracle. In this context, we will rely on a generic result from [KX22]. To this end, we first recast PROV as an instantiation of the Hash-and-Sign paradigm tailored to the UOV algorithm, as presented in Algorithm 1, composed with a variant of the BUFF generic transformation where messages are always prefixed by a hash of the public key.

Algorithm 5 The probabilistic HaS paradigm with retry.

```
procedure HaS[T, $\mathcal{H}$].KeyGeneration($1^\lambda$)
    $(F, I) \leftarrow$ Gen($1^\lambda$)
    return $(F, I)$

procedure HaS[T, $\mathcal{H}$].Verify(F, msg, (salt, $\mathbf{s}$))
    return $F(\mathbf{s}) \overset{?}{=} \mathcal{H}(\mathsf{msg}||\mathsf{salt})$

procedure HaS[T, $\mathcal{H}$].Sign(I, msg)
    $\mathbf{v} \leftarrow I^1()$
    repeat
        $\mathsf{salt} \leftarrow_{\$} \{0,1\}^{\mathsf{len}_{\mathsf{salt}}}$
        $\mathbf{s} \leftarrow I^2(\mathcal{H}(\mathsf{msg}||\mathsf{salt}))$
    until $\mathbf{s} \neq \perp$
    return $(\mathsf{salt}, \mathbf{s})$
```

One has the following generic theorem.

Theorem 2 ([KX22], Proposition 4.1)**.** *For any quantum* EUF-CMA *adversary $\mathcal{A}$ of* $\mathsf{HaS}[\mathsf{T},\mathcal{H}]$ *issuing at most q_s classical queries to the signing oracle and q_h quantum random oracle queries to $\mathcal{H}$, there exist an* INV *adversary $\mathcal{B}$ and a* PS *adversary $\mathcal{C}$ against* T *issuing q_s sampling queries such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HaS}[\mathsf{T},\mathcal{H}]}(\mathcal{A}) \le{}& (2q_h+1)^2\mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{PS}}_{\mathsf{T}}(\mathcal{C})\\ &+ \frac{3}{2}q_s'\sqrt{\frac{q_s'+q_h+1}{|\mathcal{R}|}} + 2(q_s+q_h+2)\sqrt{\frac{q_s'-q_s}{|\mathcal{R}|}},\end{aligned}$$

where q_s' is a bound on the total number of queries to $\mathcal{H}$ in all the signing queries, and the running time of $\mathcal{B}$ and $\mathcal{C}$ are about that of $\mathcal{A}$.

As per [CDF$^+$21], the BUFF transformation has no impact on the EUF-CMA security of the transformed scheme (both in the classical and quantum setting). Hence, we can simply apply Theorem 2 to PROV, as discussed in the proof of [KX22, Proposition 5.3]. Moreover, as stated in Remark 1, $q_s' > q_s$ with probability at most

$$\frac{q_s}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + \frac{q_s}{|\mathbb{F}|^{\delta}(|\mathbb{F}|-1)}.$$

Thus, one gets the following corollary.

Corollary 2. *For any quantum* EUF-CMA *adversary $\mathcal{A}$ of* PROV *issuing at most q_s classical queries to the signing oracle and q_h quantum random oracle queries to $\mathcal{H}$, there exist an* INV *adversary $\mathcal{B}$ such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{HaS}[\mathsf{T},\mathcal{H}]}(\mathcal{A}) \leq & (2q_h+1)^2 \mathsf{Adv}^{\mathsf{INV}}_{\mathsf{T}_{\mathsf{PROV}}}(\mathcal{B}) \\ & + \frac{3}{2} q_s \sqrt{\frac{q_s+q_h+1}{2^{\mathsf{len}_{\mathsf{salt}}}}} + \frac{q_s}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + \frac{q_s}{|\mathbb{F}|^{\delta}(|\mathbb{F}|-1)},\end{aligned}$$

where the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

3.2 Resistance to known cryptanalysis (2.B.5)

Direct attacks. In a forgery attack, the adversary tries to directly invert the public quadratic map. We estimate the complexity of this approach as solving a uniformly random quadratic map from $\mathbb{F}^{\mathsf{n}}$ to m. In this case, the most efficient known algorithm is the so-called hybrid attack that tries to guess k variables, and then attempts to inverse the resulting quadratic map, whose complexity is given by the formula [BFP10]:

$$\min_k \left(3|\mathbb{F}|^k \binom{\mathsf{n}-k+d_{\text{reg}}}{d_{\text{reg}}}^2 \binom{\mathsf{n}-k}{2} \right),$$

where d_{reg} is the smallest integer d so that the coefficient of z^d in

$$\frac{(1-z^2)^{\mathsf{n}}}{(1-z)^{\mathsf{n}-k}}$$

is non-positive. This estimate is based on the XL-Wiedemann solver. In order to get an accurate evaluation of the hardness of solving this system, we rely on the automatic tool by Bellini et al. in [BMSV22] that also includes other solvers.

Quantum direct attacks. We take into account the quantum version of the direct attack from [SW16, FHK$^+$17]. Its complexity for solving quadratic systems of e equations in e variables over $\mathbb{F}_2$ is $O(2^{0.462e})$ quantum gates.

Moreover, we also consider the hybrid attack from the previous paragraph, where the search part is accelerated using Grover's algorithm. Its complexity is

$$\min_k \left(3|\mathbb{F}|^{k/2} \binom{\mathsf{n}-k+d_{\text{reg}}}{d_{\text{reg}}}^2 \binom{\mathsf{n}-k}{2} \right),$$

where d_{reg} is defined as above.

Kipnis-Shamir attack. In this attack [KS98], the adversary tries to directly recover the oil space by combining on two quadratic forms. This attack targeted the original UOV scheme, where $\mathsf{n} = 2\mathsf{m}$, and $\delta = 0$. Since then, it has been extended to an attack with a complexity in $\widetilde{O}(|\mathbb{F}|^{v-o})$, where v and o respectively denote the number of oil and vinegar variables [KPG99]. In the case of PROV and ignoring polynomial factors, this attack has a complexity of $|\mathbb{F}|^{\mathsf{n}-2\mathsf{m}-2\delta}$.

Intersection attack. In this attack [Beu21], the adversary tries to recover k vectors in vector spaces of the form $\mathbf{M}_i \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix} \cap \mathbf{M}_j \begin{pmatrix} \mathbf{O} \\ \mathbf{I}_{\mathsf{m}+\delta} \end{pmatrix}$, where $\mathbf{M}_i$ and $\mathbf{M}_j$ are the matrices of the polar forms associated to the i-th and j-th components of the public quadratic map, and both matrices are invertible. This attack essentially corresponds to solving a random system of

$$\binom{k+1}{2}^2 o - 2\binom{k}{2}$$

equations in $k(v+o) - (2k+1)o$ variables. In the case of PROV, this corresponds to a system of

$$\binom{k+1}{2}^2 (\mathsf{m}+\delta) - 2\binom{k}{2}$$

equations in m variables. We follow the strategy of [BCH$^+$23] to find an optimal parameter k.

Security of the symmetric primitives. Our construction relies of symmetric primitives such as keyed and unkeyed hash functions. No attack more efficient than brute force is known against the chosen algorithms.

Side-channel attacks. While PROV was not specifically designed for resistance against side-channel attacks, we expect it to provide a good level of security in such scenarios. Indeed, contrary to the SaltedUOV construction, our choice of parameters implies that it is unlikely for the signature algorithm to need more than a single salt sampling (it happens with probability $\approx |\mathbb{F}|^{-\delta-1}$). This, along with the fact that signature generation and verification involve the evaluation of symmetric primitives and linear algebra over $\mathbb{F}$, will allow efficient masked implementations of PROV. Moreover, as we reuse the field of AES, we can benefit from various masked implementation for the multiplication.

Other attacks. In [CDF$^+$21], Cremers et al. introduce the BUFF generic construction that, given an EUF-CMA-secure signature schemes, builds a new signature algorithm that also satisfies the following security notions:

- exclusive ownership [PS05]: a signature should verify only under a single public key;
- message-bound signature: a signature should only be valid for a single message;
- non re-signability [JCCS19]: one should not be able to produce a signature under another key given a signature for an unknown message.

This transformation simply hashes the public key along with the message to be signed in the signature generation and verification algorithm. Given the size of our public keys, this would entail a significant performance overhead. Hence, we chose to instead rely on a hash of the public key. Under the assumption that the underlying hash function is collision-resistant, this is sufficient to guarantee the same additional security guarantees as the BUFF transformation.

3.3 Expected security of parameter sets (part of 2.B.4)

As seen in Section 3, PROV can be proven secure under the assumption that the $\mathsf{UOV}^{\text{-}}$ problem is hard to solve. This bounds the possible information leakage about the secret key by the signing oracle. Our parameter selection was in part guided by the bound from Corollary 1. More concretely, we have the following criterion:

$$\frac{q_s}{|\mathbb{F}|^{\mathsf{n}-\mathsf{m}-\delta}} + \frac{q_s}{|\mathbb{F}|^{\delta}(|\mathbb{F}|-1)} + \frac{q_s(q_h+q_s)}{2^{\mathsf{len}_{\mathsf{salt}}}} \lesssim 1, \tag{6}$$

when $q_s \leq 2^{64}$ and $q_h \leq 2^{\lambda}$. However, due to the $(q_s + q_h + 1)$ factor in the reduction to the $\mathsf{UOV}^{\text{-}}$ inversion problem, we would need to ensure 2λ bits of security for the underlying INV problem in order to provably guarantee λ bits of security for PROV. Since this would come with a prohibitive performance cost, we chose to ensure at least λ bits of security for the underlying INV problem. This seems reasonable as multivariate cryptography schemes have thus far never been attacked through the information leakage of their signature algorithm. As far as quantum security is concerned, Corollary 2 only provides asymptotic security for PROV. Hence, we rely on the criterion from Equation (6) and on the analysis from Section 3.2. Concrete parameters for NIST levels I, III and V can be found in Table 1.

3.4 Practical estimates (part of 2.B.4)

The analysis derived from the security proof in Section 3.3 mainly serves to choose the size of δ, and the salt length, in relation to the proof. It remains to assess the practical performance of the various attacks presented earlier in this section against our parameter choices. The results are depicted on Figure 2.

The first two attacks are direct attacks. The system is solved either using the XL Wiedemann algorithm, estimated using the same formula as the NIST Rainbow submission. The second line shows the hybrid F5 solver, computed using an automatic tool by Bellini et al. in [BMSV22]. In both cases, the value k in parenthesis shows the optimal number of guessed variables for the algorithm.

The last two attacks are attacks against the UOV problem. We have been especially conservative in our choice of parameters with regard to those attacks, since the hardness of the UOV problem is not as well-established as the hardness of the MQ problem. For the intersection attack, the value of k indicates the optimal number of intersected spaces.

In other words, any distinguishing attack against PROV implies a distinguishing attack against an UOV system with m equations in $\mathsf{n} + \delta$ variables.

On the practical side, all attacks estimate the number of field operations, not the number of gates. The number of gates for each field operations can be estimated as $2\log(|\mathbb{F}|)^2 + \log(|\mathbb{F}|)$, which would add around 8 bits of security. We have chosen not to take this fact into account as an extra caution, but also because we believe the number of field operations is more representative of the complexity of classical attacks, especially on $\mathbb{F}_{2^8}$.

	PROV-I	PROV-III	PROV-V
λ	128	192	256
XL-Wiedemann	137 $(k = 1)$	203 $(k = 5)$	263 $(k = 5)$
Hybrid F5	**131** $(k = 1)$	**197** $(k = 3)$	**257** $(k = 5)$
Kipnis-Shamir	238	335	464
Intersection	172 $(k = 2)$	234 $(k = 2)$	315 $(k = 2)$

Table 2: Complexity of the main attacks against PROV.

4 Advantages and limitations (2.B.6)

4.1 Advantages

Simplicity. Like most UOV-based signature schemes, PROV has a very simple design that is both easy to understand and to implement. Indeed, the only operations that are required in order to generate and verify a signature are matrix multiplications and solving linear systems of equations over the finite field $\mathbb{F}$.

Provable security. PROV can be proven secure both in the ROM and the QROM, under the assumption that the UOV$^-$ problem is hard to solve. Moreover, this hardness assumption has been well-studied since the publication of UOV in [KPG99].

It is worth noting that distinguishing a PROV system with from a uniformly random system of equations is strictly harder than distinguishing a corresponding UOV cryptosystem with with m equations in $\mathsf{n} + \delta$ variables. Indeed, the attacks has access to strictly less information (due to δ missing equations). This point is valid in practice: our parameter sets are expected to resist attacks even if the number of equations was increased to $\mathsf{m} + \delta$.

Short signatures. As is often the case in multivariate cryptography (see e.g.[Beu22, BCH$^+$23, FIKT21]), one of the main selling points of PROV is its small signature size. This is especially true when compared to MPC-in-the-head schemes that are directly based on the Multivariate Quadratic problem whose signatures are larger than 10KB for a security level of 128 bits (see e.g. [CHR$^+$16, Beu20]).

4.2 Limitations

Key sizes. Like most other multivariate schemes, the main limitation of PROV is its relatively large public key and expanded secret key size. Moreover, in order to provide a meaningful security proof, we had to increase the dimension of the oil space. Hence, this also lead to a corresponding increase in the number of vinegar variables, which increased both the signature size and the public key sizes when compared with other constructions based on UOV such as [BCH$^+$23].

References

[BCH$^+$23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. TCHES 2023, To appear, 2023. `https://eprint.iacr.org/2023/059`.

[Beu20] Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211. Springer, Heidelberg, May 2020.

[Beu21] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 348–373. Springer, Heidelberg, October 2021.

[Beu22] Ward Beullens. MAYO: Practical post-quantum signatures from oil-and-vinegar maps. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021*, volume 13203 of *LNCS*, pages 355–376. Springer, Heidelberg, September / October 2022.

[BFP10] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Crypt*, 2:1–22, 01 2010.

[BMSV22] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier A. Verbel. An estimator for the hardness of the MQ problem. In Lejla Batina and Joan Daemen, editors, *AFRICACRYPT 22*, volume 2022 of *LNCS*, pages 323–347. Springer Nature, July 2022.

[BPB10] Stanislav Bulygin, Albrecht Petzoldt, and Johannes Buchmann. Towards provable security of the unbalanced Oil and Vinegar signature scheme under direct attacks. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 17–32. Springer, Heidelberg, December 2010.

[CDF$^+$21] Cas Cremers, Samed Düzlü, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. In *2021 IEEE Symposium on Security and Privacy*, pages 1696–1714. IEEE Computer Society Press, May 2021.

[CHR$^+$16] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-pass MQ-based identification to MQ-based signatures. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 135–165. Springer, Heidelberg, December 2016.

[FHK$^+$17] Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast quantum algorithm for solving multivariate quadratic equations. *CoRR*, abs/1712.07211, 2017.

[FIKT21] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. A new variant of unbalanced Oil and Vinegar using quotient ring: QR-UOV. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 187–217. Springer, Heidelberg, December 2021.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[JCCS19] Dennis Jackson, Cas Cremers, Katriel Cohn-Gordon, and Ralf Sasse. Seems legit: Automated analysis of subtle attacks on protocols that use signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2165–2180. ACM Press, November 2019.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Heidelberg, May 1999.

[KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil & Vinegar signature scheme. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 257–266. Springer, Heidelberg, August 1998.

[KX22] Haruhisa Kosuge and Keita Xagawa. Probabilistic hash-and-sign with retry in the quantum random oracle model. Cryptology ePrint Archive, Report 2022/1359, 2022. `https://eprint.iacr.org/2022/1359`.

[Lev05] A. A. Levitskaya. Systems of random equations over finite algebraic structures. *Cybernetics and Systems Analysis*, 41:67–93, 2005.

[PBB10] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. CyclicRainbow - a multivariate signature scheme with a partially cyclic public key. In Guang Gong and Kishan Chand Gupta, editors, *INDOCRYPT 2010*, volume 6498 of *LNCS*, pages 33–48. Springer, Heidelberg, December 2010.

[PS05] Thomas Pornin and Julien P. Stern. Digital signatures do not guarantee exclusive ownership. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 138–150. Springer, Heidelberg, June 2005.

[PTBW11] Albrecht Petzoldt, Enrico Thomae, Stanislav Bulygin, and Christopher Wolf. Small public keys and fast verification for $\mathcal{M}$ultivariate $\mathcal{Q}$uadratic public key systems. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 475–490. Springer, Heidelberg, September / October 2011.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of UOV and HFE signature schemes against chosen-message attack. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, pages 68–82. Springer, Heidelberg, November / December 2011.

[SW16] Peter Schwabe and Bas Westerbaan. Solving binary MQ with grover's algorithm. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2016.