

VOX

Specification v1.1 – 05/17/2024

Principal submitter: Jacques Patarin
University of Versailles-St-Quentin-en-Yvelines
LMV Laboratory
45 avenue des Etats-Unis
FR-78035 Versailles Cedex
France
Thales DIS France
6 rue de la verrerie
92190 Meudon
France
jacques.patarin@thalesgroup.com
phone: +33 6 28 34 74 58

Auxiliary submitters: Benoît Cogliati
Jean-Charles Faugère
Pierre-Alain Fouque
Louis Goubin
Robin Larrieu
Gilles Macario-Rat
Brice Minaud

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: vox-sign.com

Alternative point of contact: 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

Contents

1	Introduction	3
1.1	General introduction	3
1.2	Design rationale and comparison with other UOV schemes	5
2	General algorithm specification (part of 2.B.1)	6
2.1	Parameter space	6
2.2	Secret-key and public-key	6
2.3	Signing process	8
2.4	Verification process	10
3	List of parameter sets (part of 2.B.1)	11
4	Design rationale (part of 2.B.1)	11
5	Detailed performance analysis (2.B.2)	12
5.1	Benchmarks	12
5.2	Implementation details	12
6	Expected strength (2.B.4) for each parameter set	16
7	Analysis of known attacks (2.B.5)	18
7.1	Direct signature forgery attacks	18
7.2	Kipnis-Shamir attack	18
7.3	Intersection attack	18
7.4	Quantum analysis	18
7.5	The Rectangular Minrank Attack	19
8	Advantages and limitations (2.B.6)	19

History of the document

Version	Date	Comments
1.0	01/06/2023	Initial specification of the VOX family, including a specific instantiation called “VOX-F” (or Full VOX, nicknamed “FOX”) that does not use the QR technique and allows to obtain easier security arguments.
1.0a	28/07/2023	Clarification added in the matrix representation of the linear mappings and their operations on vectors.
1.1	17/05/2024	In-depth analysis of the impact of rectangular attacks [FI23][INT23][TPD21] and other attacks [GD24, Péb24b, Péb24a] led us to abandon the “Quotient Ring” (QR) structure, so that rank defects can no longer be exploited. This new specification focuses on “VOX-F” (or Full VOX, nicknamed “FOX”) with optimized parameters for the targeted I, III, V security levels, and a detailed performance analysis.

1 Introduction

FOX is a UOV-based hash-and-sign signature scheme from the Multivariate Quadratic (MQ) family. It uses a new variant proposed by Faugère, Macario-Rat, Patarin, and Perret under the name $\text{UOV}\hat{+}$. The $\text{UOV}\hat{+}$ has an interesting security property. The quadratic forms of a UOV public key have a large common isotropic subspace, which is unusual for a random quadratic system. $\text{UOV}\hat{+}$ adds a small number of random quadratic forms to the public keys to hide this subspace. This makes the signing process more complex since we have to solve a small quadratic system, but also allows us to reduce the public key of UOV for equivalent security.

Originally FOX was part of a larger family of signature schemes, submitted in June 2023 at NIST call for public key signature schemes, under the name VOX. The submitters have decided to keep track of only the FOX instantiation, hence VOX still denotes the submission itself, whereas the present specification and also the web site (<https://vox-sign.com/>) are now focused on our main candidate: FOX.

1.1 General introduction

Multivariate Cryptography is a mature proposal in the landscape of public-key alternatives to RSA and Elliptic Curve Cryptography since 1988 and the introduction of the Matsumoto-Imai cryptosystem. Many encryption and signature schemes have been proposed in the nineties and Unbalanced Oil and Vinegar (UOV for short) is one interesting example. One of the main advantages of this family is to propose very small signature size, which is one important feature of the new NIST call for proposal. Indeed, hash-based signatures and lattice-based signatures are very large, and MQ can be a good intermediate for short signatures. The main drawback of this kind of scheme is the size of the public-key, but fortunately, many new ideas allow to reduce the public-key size.

VOX is a family of multivariate post-quantum public key signature scheme. Its name means “Vinegar, Oil and Multiplications”, or alternatively “Extended Oil and Vinegar Signature scheme”. As we will see its main advantages are:

- it generates very short signatures,
- fast verification of signatures,
- its simplicity,

- a public key of reasonable size (which is uncommon in multivariate cryptography),
- it is based on UOV, a well-known scheme studied for more than 20 years,
- FOX has strong security arguments.

General Presentation of FOX. We start by quickly presenting the 3 main ideas behind the design of FOX. A detailed mathematical presentation of FOX will be given in other sections of this document, with more explanations.

1. **The UOV algorithm.** This algorithm was presented at EUROCRYPT 1999 in [KPG99] and has been the subject of many research papers since 24 years. UOV is therefore a well-known mature algorithm, and moreover it has a very simple design.

The general principle is as follows: we will have two kinds of variables, o “oil” variables, and v “vinegar” variables. We also have a secret system of o quadratic equations in these $(o + v)$ variables such that we never have a product of an oil variable with another oil variable. Moreover the public key is obtained with a secret linear change of variable that will mix the oil and vinegar variables. To sign a message, we start by randomly fixing the vinegar variables. Hence, we obtain a linear system of o equations in the o oil variables that can then easily be solved.

2. **The BPB technique.** This technique (see [PBB10, BPB10, PTBW11]) improves the key generation step by significantly reducing the public key size without any security degradation. Since in multivariate cryptography the size of the public key is often a bit large, this technique is very appreciated. Besides, its impact on the performance is small as it only involves solving linear equations.
3. **The $\hat{+}$ technique.** This is a relatively new technique, presented in [FmRPP22] by Faugère *et al.* It is one of the main innovations of FOX compared with UOV.

The idea is to mix a small number t of random quadratic equations in the initial set of secret quadratic equations. (This explains the “X” letter in FOX since we will add some oil times oil terms). Since the t equations are truly random it is reasonable to think that this process can only increase the security. However, in order to sign a message we will now have to solve a quadratic system in t variables (after a gaussian elimination). This can be done in an efficient way (for example with Gröbner basis variants) thanks to the fact that t is small (t will be typically between 1 and 9 for example). One of the beauty of this idea is the fact that the complexity of this resolution depends on t but is almost independent of the size of the finite field the variables belong to (we do not perform exhaustive search on t values), unlike the complexity of the attacks that seems to increase rapidly with this size.

A short history of UOV and FOX. The “Oil and Vinegar” algorithm was first presented in September 1997 (see [Pat97]). At that time the number of vinegar variables was the same as the number of oil variables ($o = v$). Due to this reason, the scheme was broken in 1998 by Kipnis and Shamir in [KS98]: they were able to recover the secret key.

Soon after, it was noticed that this attack does not work when v is sufficiently larger than o . This gives the UOV algorithm (see [KPG99]). Since 1999, the UOV signature scheme has not been broken, and some ideas have been found in order to reduce its public key size in [PBB10, BPB10, PTBW11].

UOV satisfies very good properties (simplicity of the design, short signatures, no known efficient quantum attacks, speed of signature generation and verification), it may seem weird that UOV was not already submitted in the first NIST call for PQ signature schemes. The explanation is the following: in fact two variants of UOV were submitted, Rainbow and

LUOV. Rainbow [DS05] is a 2005 variant where one uses more than 2 types of variables, in order to obtain very short signature. LUOV [BP17] is a 2017 variant where we use matrices on a sub-field, in order to reduce the public key size. However these two variants were found to be disappointing due to the discovery of new attacks: see [Beu22a] on Rainbow, and [DDS⁺20] on LUOV. Nevertheless, a precise analysis of these new attacks shows that the underlying UOV problem is still difficult, and therefore is still an efficient scheme when its parameters are well chosen [BCH⁺23].

As said in introduction, FOX was first a specific instantiation described in the original VOX submission. In this submission, an additional technique known as the QR (for Quotient Ring) technique, originally presented in [FIKT21], was part of the features. FOX can be considered as a VOX variant in which the QR technique has been omitted. Indeed, recent attacks against VOX variants that use the QR technique were published, which led the submitters to keep only the FOX family.

Some arguments of security. Finding a solution of a general system of multivariate quadratic equations (MQ problem) over a finite field K is an NP-hard problem (for every finite field K). Moreover, this problem seems to have the same complexity on average and in the worse case (however this is not proved). For random quadratic equations, the best known algorithms combine Gröbner bases variants and exhaustive search of some variables. Their complexities are well studied, and often increase exponentially with the number of variables.

The quadratic systems used in UOV and FOX are not random (since there is a trapdoor) and we do not claim that solving these kind of systems is NP-hard. The MinRank problem (used in the cryptanalysis of many multivariate systems to recover the secret key) is (as MQ) an NP-hard problem. Due to its simplicity, the UOV problem has its own interest. As we will see later there is a proof of EUF-CMA security for UOV based on this problem [SSH11], which shows that computing UOV signatures does not give information on the secret key.

Side-Channel Attacks. It is expected that all cryptographic algorithms can be attacked with side-channel attacks. However it is also expected that it is possible to obtain a sufficient practical security by using countermeasures, typically by using appropriate software techniques. UOV and FOX are not an exception.

Known side-channel attacks on UOV (see for example [FKNT22, ACK⁺23]) are not very impressive (more details will be given later in this document). Moreover it seems relatively easier to write countermeasures on UOV/FOX against side channel attacks than for most of other public key algorithms. This comes from the fact that in UOV/FOX the secret key is relatively large (if a few bits of the secret key are found the scheme may remain secure) and also from the fact that the techniques to protect the linear secret transformation are relatively easy and classical. However, side-channel attacks on FOX are not the main subject of this document.

1.2 Design rationale and comparison with other UOV schemes

Our main objective is to propose a signature scheme with small signature size, since this is the drawback of lattice-based and hash-based signature, and with alternative security assumption. In the same time, FOX introduces new ideas to reduce the size of the public key, which is an important parameter for bandwidth-constraint applications, which is one of the drawback of post-quantum schemes and also for MQ schemes. We think that the UOV⁺ improves the security of UOV, which makes FOX close to a random quadratic system. The inherent structural property of UOV, having a common isotropic subspace for all quadratic forms of the public key is no more present.

As a consequence, FOX-I achieves signature of size 124B with public key of size 50KB. The performances of key generation, signing and verification are rather good compared to other schemes. In the following, we compare FOX with other attractive UOV signature schemes.

The UOV Signature scheme [KPG99]. UOV has a public-key of $O(mn^2 \log q)$ bits for m quadratic equations in n variables. For a 128-bit security level (NIST Level 1), we can take $m = 53, n = 3m$ and $q = 31$, which amounts to 421 KB. With BPB trick, the key size goes from $O(mn^2 \log q)$ bits to $O(m^3 \log q)$ bits, and the public key size is 48KB.

Since the signature size is $O(n \log q)$, this is 100B. The signature size of UOV is very short.

A recent paper by Beullens et al. [BCH⁺23] gives similar secure parameters for UOV: “At NIST security level 1, this results in either 128-byte signatures with 44 kB public keys or 96-byte signatures with 67 kB public keys.”

The MAYO Signature [Beu22b]. For 128 security bits (NIST Level 1), we can take $m = 60, n = 62$ and $q = 31$, and an oil space of dimension $o = 6$, which is 420B for the signature and 803B for the public-key. The improvement of MAYO is the size of the public-key from $m^3 \log q$ to $mo^2 \log q$, with a new security assumption since the size of the oil space is small. This allows to reduce the number of variables. However, it adds a novel security assumption, called “Whipped MQ Problem”, and defined in page 13 of the paper. In conclusion, MAYO achieves parameters close to Falcon for the signature and public key size, under MQ-related assumptions.

MPC-in-the-head signature schemes. MQ-based MPC in the head signature schemes offer security proofs under the MQ hypothesis, without the need of an underlying trapdoor. However, these high guarantees come at the cost of high signature and public key sizes. As an example, the MQDSS scheme [CHR⁺16, CHR⁺20] has a signature size of 27 KB for a security level of 128 bits. More recent schemes such as MUDFISH and SUSHY-FISH [Beu20b] achieve smaller signature sizes of respectively 14 KB and 12 KB for the same security level.

2 General algorithm specification (part of 2.B.1)

2.1 Parameter space

The main parameters involved in FOX are:

- λ , the security level of FOX, with possible values 128, 192, and 256,
- q , a prime power, order of a finite field \mathbb{F}_q ,
- o , the number of Oil variables, also the number of equations in public and secret key,
- v , the number of Vinegar variables,
- $t \leq o$, the number of totally random equations in the secret key.

2.2 Secret-key and public-key

The public-key in FOX is a set $\mathbf{Pub} = \{p_1, \dots, p_o\} \in \mathbb{F}_q[x_1, \dots, x_{o+v}]$ of o homogeneous quadratic equations in $o + v$ variables. So formally,

$$p_i = \sum_{1 \leq k \leq j \leq o+v} a_{i,j,k} x_j x_k \quad i = 1, \dots, o, \quad (1)$$

where $a_{i,j,k}$ are elements of \mathbb{F}_q . The secret-key is composed of a set $\mathbf{Sec} = \{p'_1, \dots, p'_o\} \in \mathbb{F}_q[x_1, \dots, x_{o+v}]$ of o homogeneous quadratic equations in $o + v$ variables and two random bijective linear mappings \mathbf{S}, \mathbf{T} , respectively in $\text{GL}_o(\mathbb{F}_q)$ and $\text{GL}_{o+v}(\mathbb{F}_q)$. In the secret polynomials \mathbf{Sec} , the first o variables $\{x_1, \dots, x_o\}$ are called the Oil variables, the last v variables $\{x_{o+1}, \dots, x_{o+v}\}$ are called the Vinegar variables. The polynomials of \mathbf{Sec} have the generic form of quadratic homogeneous polynomials:

$$p'_i = \sum_{1 \leq k \leq j \leq o+v} a'_{i,j,k} x_j x_k \quad i = 1, \dots, o, \quad (2)$$

where $a'_{i,j,k}$ are elements of \mathbb{F}_q ; moreover the last $o - t$ polynomials $\{p'_{t+1}, \dots, p'_o\}$ of \mathbf{Sec} have the special (UOV) form :

$$p'_i = \sum_{\substack{1+o \leq j \leq o+v \\ 1 \leq k \leq j}} a'_{i,j,k} x_j x_k \quad (3)$$

or equivalently the coefficients $a'_{i,j,k}$ are canceled for $i = t + 1, \dots, o$ and $1 \leq k \leq j \leq o$ or said otherwise, there is no “Oil×Oil” monomial in these polynomials.

By analogy, we call also the first t polynomials as the Vinegar polynomials, and the last $o - t$ polynomials as the Oil polynomials. Likewise, we call the triplets (i, j, k) (or associated monomials) satisfying $t + 1 \leq i \leq h, 1 \leq k \leq j \leq o$, the Oil triplets, and the complement, that is: $1 \leq i \leq t, 1 \leq k \leq j \leq o + v$ or $(t + 1 \leq i \leq o, 1 \leq k \leq j$ and $o + 1 \leq j \leq o + v)$, the Vinegar triplets. We note \mathbf{Pub}_v and respectively \mathbf{Pub}_o the part composed exclusively of vinegar monomials (resp. oil monomials) and \mathbf{Sec}^v , (resp. \mathbf{Sec}^o) the vinegar polynomials (resp. oil polynomials).

The public-key and the secret key are required to satisfy:

$$\mathbf{Sec} = \mathbf{S} \circ \mathbf{Pub} \circ \mathbf{T}. \quad (4)$$

Since the security and the mechanism of the scheme rely on two different types of variables and equations, the two linear mappings \mathbf{T} and \mathbf{S} may be expressed as block matrices in the following way:

$$\mathbf{S}_{o \times o} = \begin{pmatrix} I_t & \mathbf{S}'_{t \times (o-t)} \\ 0 & I_{o-t} \end{pmatrix}, \quad \mathbf{T}_{(o+v) \times (o+v)} = \begin{pmatrix} I_o & \mathbf{T}'_{o \times v} \\ 0 & I_v \end{pmatrix}, \quad (5)$$

where I is the identity matrix with given dimension, $\mathbf{S}'_{t \times (o-t)}$ and $\mathbf{T}'_{o \times v}$ are rectangular full random block matrices with given dimensions. Note that due to the presence of identity blocks and their triangular form, the matrices of \mathbf{S} and \mathbf{T} are invertible, whatever the values of \mathbf{T}' and \mathbf{S}' .

Note also that for convenience we use the same symbol for a linear mapping and its matrix representation. However, we use the line vector convention, therefore for instance $(p_1, \dots, p_o) = \mathbf{S}(p'_1, \dots, p'_o) = (p'_1, \dots, p'_o) \times \mathbf{S}$ and $\mathbf{T}(x_1, \dots, x_{o+v}) = (x_1, \dots, x_{o+v}) \times \mathbf{T}$. Vectors are multiplied on the left of the matrices.

2.2.1 The BPB Compression

This feature has been introduced in [PBB10, BPB10]. Its purpose is also to decrease the size of the public key. It has no impact on the security, and almost no impact on the generation time. The idea of this feature is that part of the public key can be stored as the seed of DRBG, while the secret key still can keep its property of randomness.

The equation $\mathbf{Sec} = \mathbf{S} \circ \mathbf{Pub} \circ \mathbf{T}$ says that the coefficients of \mathbf{Sec} are polynomial expressions in the coefficients of \mathbf{S} , \mathbf{Pub} , and \mathbf{T} , with respectively degrees 1, 1, and 2. Therefore, when \mathbf{S} and \mathbf{T} are set, then the coefficients of \mathbf{Sec} are linear expressions in

the coefficients of \mathbf{Pub} and vice versa. The idea developed in [PBB10, BPB10], says that you can fix the Vinegar coefficients of \mathbf{Pub} to random values, and find the Oil coefficients of \mathbf{Pub} , satisfying a linear system stating that the Oil coefficients of \mathbf{Sec} must be zeros. Practically, if \mathbf{Pub}_v and \mathbf{Pub}_o are the Vinegar and Oil parts of the public key, they must satisfy

$$(\mathbf{S} \circ \mathbf{Pub}_v \circ \mathbf{T})_o + (\mathbf{S} \circ \mathbf{Pub}_o \circ \mathbf{T})_o = 0.$$

However, due to the special triangular block forms of \mathbf{S} and \mathbf{T} , we have

$$(\mathbf{S} \circ \mathbf{Pub}_o \circ \mathbf{T})_o = \mathbf{Pub}_o.$$

Therefore in our case, the BPB compression simply amounts to set:

$$\mathbf{Pub}_o = -(\mathbf{S} \circ \mathbf{Pub}_v \circ \mathbf{T})_o.$$

2.2.2 Key generation process

The overall key generation process is detailed in alg. 1.

Algorithm 1 Key Generation for Compressed FOX public keys

```

1: procedure CRYPTOSIGNKEYPAIR(sk, pk)
2:   SeedSec  $\leftarrow$  randombytes(32)
3:    $\mathbf{S} \leftarrow$  GenerateS(SeedSec)
4:    $\mathbf{T} \leftarrow$  GenerateT(SeedSec)

5:   SeedPub  $\leftarrow$  randombytes(32)
6:    $\mathbf{Pub}_v \leftarrow$  GeneratePub(SeedPub)
7:    $\mathbf{Pub}_o \leftarrow -(\mathbf{S} \circ \mathbf{Pub}_v \circ \mathbf{T})_o$ 

8:    $\mathbf{Sec} \leftarrow \mathbf{S} \circ \mathbf{Pub} \circ \mathbf{T}$ 
9:   sk  $\leftarrow$  ENCODESK(SeedSec, SeedPub,  $\mathbf{Sec}$ )
10:  pk  $\leftarrow$  ENCODEPK(SeedPub,  $\mathbf{Pub}_o$ )
11: end procedure

```

2.2.3 Key material encoding

The generation process in alg. 1 shows that each element can be stored and or used in a compact and or expanded form.

The maximum compressed format for the secret key is the concatenation of SeedSec and SeedPub, hence 64 bytes.

The maximum compressed format for the public key is the concatenation of SeedPub and a compact representation of \mathbf{Pub}_o , hence $32 + \lceil ((o-t) * o * (o+1)) / 2 * \lceil \log_2(q) \rceil / 8 \rceil$ bytes.

An intermediate format for the secret key is to store SeedPub, SeedSec and \mathbf{Sec} , which is a good tradeoff between time and memory.

In FOX, q has been chosen as the nearest prime by default to a power of two, as to minimize the loss of memory for a compact representation. Therefore it is efficient to store a list of elements of \mathbb{F}_q as packed bits. Seeds can naturally be stored as byte strings.

2.3 Signing process

FOX uses the Hash and Sign paradigm. To sign a message M , FOX requires a hash function H valued in the arrival set of \mathbf{Pub} , which is namely $(\mathbb{F}_q)^o$. A signature σ of the message M is a solution in x of the equation $\mathbf{Pub}(x) = H(M)$. In order to solve it, FOX uses the

knowledge of the secret key and solve instead the equation in z : $\mathbf{Sec}(z) = \mathbf{S}(H(M))$, then a signature is $\sigma = \mathbf{T}(z)$.

Solving $\mathbf{Sec}(z) = y$ uses the special form of its polynomials and the same kind of hint used for UOV. First sets the vinegar variables to some random values. (At this point we made the choice to initiate a DRBG with a seed that depends on the secret key, the message and a counter. The counter takes into account the attempts without solutions.) Then the system can be split into two parts : an affine part and a quadratic part. The affine part is a system of $o - t$ equations in o variables. This part can be used to eliminate $o - t$ variables, that is $o - t$ (say) first variables can be expressed as affine combinations of the (say) last t variables¹. Then these $o - t$ variables are eliminated from the the quadratic part, that is these variables are replaced by their expression in the last t variables. Then the quadratic part becomes a system of t quadratic equations in t variables. This system can be solved in a classic way by computing a (graded reverse lexicographic) Gröbner basis, then a (lexicographic) Gröbner basis, then solving amounts to find the roots of one univariate polynomial. This in turn gives the possible values of the t variables, then the other $o - t$ variables follow by using their affine expressions. Adding the initial random values of the vinegar variables gives a complete solution for z . Multiplying z by \mathbf{T} gives the signature.

When the system has more than one solution, we need a procedure to chose deterministically one of them. Our choice is to draw at random at the same time the Vinegar is drawn, one supplementary value used as a milestone. We then consider \mathbb{F}_q as a dial of a clock with q values, then the chosen solution is the first one encountered on the dial when starting from the milestone and traveling clockwise. This procedure is simple and was driven by the fact that it reuses the random generator of one element amongst q , however it lacks of uniformity (for instance when there are two solutions close one of another, the first one will be chosen more often than the other). If the lack of uniformity is an issue, we will replace this procedure by a more straightforward choice at random of one solution amongst all.

¹For a random choice of the Vinegar variables, this affine system has a probability $1/q^{t+1}$ to be singular.

Algorithm 2 Signature Algorithm

```

1: function FOXSIGN(hash, SeedVinegar, S, T, Sec)
2:    $y \leftarrow \text{hash} \times \mathbf{S}$ 
3:   count  $\leftarrow 0$ 
4:   repeat
5:     repeat
6:       vinegar, hint  $\leftarrow \text{GENERATEVINEGAR}(\text{SeedVinegar}, \text{count})$   $\triangleright$  Deterministic
       random vinegar + milestone for choosing solution
7:       count  $\leftarrow \text{count} + 1$ 
8:        $L \leftarrow \text{EVALUATE}(\mathbf{Sec}^o - y_o, \text{vinegar})$ 
9:       success,  $H \leftarrow \text{SOLVELINEAR}(L)$ 
10:      until success  $\triangleright$  here,  $L$  is regular
11:       $Q \leftarrow \text{EVALUATE}(\mathbf{Sec}^v - y_v, \text{vinegar}, H)$ 
12:      success,  $\xi \leftarrow \text{SOLVEQUADRATIC}(Q, \text{hint})$ 
13:      until success  $\triangleright$  here,  $Q$  is regular and has a solution
14:      oil  $\leftarrow H \times \xi$ 
15:       $z \leftarrow \text{vinegar} \parallel \text{oil}$ 
16:       $\sigma \leftarrow z \times \mathbf{T}$ 
17:      return  $\sigma$ 
18: end function

19: procedure CRYPTOSIGN(SignedMessage, Message, sk)
20:   SeedSec, SeedPub, T, S, Sec  $\leftarrow \text{DECODESK}(\text{sk})$ 
21:   hash, SeedHash  $\leftarrow \text{HASHANDSEED}(\text{Message}, \text{SeedPub})$ 
22:    $\sigma \leftarrow \text{FOXSIGN}(\text{hash}, \text{SeedSec} \parallel \text{SeedHash}, \mathbf{S}, \mathbf{T}, \mathbf{Sec})$ 
23:   SignedMessage  $\leftarrow \text{ENCODESIGNATURE}(\text{Message}, \sigma)$ 
24: end procedure

```

2.4 Verification process

The verification process is simple. To verify a signature σ of a message M , it suffices to compute $\mathbf{Pub}(\sigma)$ and $H(M)$, and check that these values are equal. See alg. 3.

Algorithm 3 Verification Algorithm

```

1: function FOXVERIFY( $\sigma$ , hash, Pub)
2:   Result  $\leftarrow \text{true}$ 
3:   for  $i \leftarrow 1, o$  do
4:     Result  $\leftarrow \text{Result AND } (\sigma_i \text{ EQUAL EVALUATE}(p_i, \text{hash}))$ 
5:   end for
6:   return Result
7: end function

8: function CRYPTOSIGNOPEN(SignedMessage, Message, pk)
9:   SeedPub, Pub  $\leftarrow \text{DECODEPK}(\text{pk})$ 
10:  Message,  $\sigma \leftarrow \text{DECODESIGNEDMESSAGE}(\text{SignedMessage})$ 
11:  hash,  $\perp \leftarrow \text{HASHANDSEED}(\text{Message}, \text{SeedPub})$ 
12:  return FOXVERIFY( $\sigma$ , hash, Pub)
13: end function

```

3 List of parameter sets (part of 2.B.1)

We define the parameters for each variant of FOX according to the three security levels defined by the NIST.

Variant	Security Level	q	o	v	t	sig	cpk	sk	csk
FOX-I	1	251	49	75	8	124 B	50,257 B	329,653 B	64 B
FOX-III	3	4093	70	106	8	186 B	183,182 B	1,103,912 B	64 B
FOX-V	5	65521	91	140	8	462 B	694,908 B	4,182,124 B	64 B

Figure 1: Parameter sets and corresponding key and signature sizes for the FOX signature scheme.

4 Design rationale (part of 2.B.1)

FOX has to be seen as a natural extension of UOV, driven by the parameter t , number of totally random equations in the secret key. The value $t = 0$ corresponds to the original UOV scheme, while $t = o$ corresponds to a complete random system of equations, with no particular “trapdoor”. Therefore the parameter t has to be adjusted between those two extreme values 0 and o , so that the tradeoff enables both security (difficulty to break the scheme) and efficiency (ability to invert the trapdoor). We can assert that, as soon as $t > 0$, FOX is stronger than UOV, with other parameters alike. For instance, the Oil-finding problem of UOV that we can state as:

Problem 1. *Given a UOV public key \mathbf{Pub} , find O , a o -dimensional subspace such that \mathbf{Pub} vanishes on O .*

becomes for FOX:

Problem 2. *Given a FOX public key \mathbf{Pub} , find O , a o -dimensional subspace such that \mathbf{Pub} on O , is a set of quadratic polynomials of dimension t .*

If one knows how to solve the FOX-problem for a generic t , then she can ipso facto solve the problem for $t = 0$.

Regarding the inversion of the trapdoor of FOX, we have shown that its core amounts to solve a quadratic system of t equations in t variables. From an attacker point of view, inverting the public system roughly amounts to solve a system of o equations in o variables. Therefore t also measures in a certain way the advantage of the legitimate user upon the attacker.

Our main argument for the security of FOX, is that any attack against it would primarily be a UOV-attack, where the attacker selects at random equations in the FOX public key-space, and hopes that by chance that they are in fact UOV-like. Indeed, if one selects a random linear combination of FOX public key polynomials, then there is one chance out of q^t , that the contribution of the vinegar polynomials (the first t ones) vanishes. However, it is obvious that any distinguishing key attack against UOV requires at least 2 equations. Therefore choosing the parameters such as $q^{2t} > 2^\lambda$ prevents any UOV-like attack.

It has been demonstrated that the BPB construction has no impact on the security of the UOV scheme and is all benefit to the construction of secret and public key, since it helps to represent them in a condensed way. It is obviously also the case for the FOX scheme.

5 Detailed performance analysis (2.B.2)

5.1 Benchmarks

We test our implementation on a 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz processor.

In the following (C) represents the reference implementation, while (AVX2) the optimized implementation.

- **FOX-I Benchmarks: average of 100 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	11.49 ms	29,771,117	13.25 ms	34,339,422
Sign	4.24 ms	11,000,844	5.07 ms	13,153,469
Sign (cached SK)	3.61 ms	9,362,493	4.34 ms	11,252,449
Verify	1.05 ms	2,723,488	1.32 ms	3,423,563
Verify (cached PK)	0.04 ms	106,974	0.06 ms	168,437

- **FOX-III Benchmarks: average of 100 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	48.33 ms	125,270,708	51.53 ms	133,580,482,
Sign	7.27 ms	18,834,197	7.90 ms	20,484,823
Sign (cached SK)	4.61 ms	11,946,513	4.94 ms	12,798,162
Verify	6.37 ms	16,516,109	6.83 ms	17,712,780
Verify (cached PK)	0.13 ms	348,334	0.13 ms	330,683

- **FOX-V Benchmarks: average of 100 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	166.38 ms	431,306,224	562.99 ms	1,459,339,050
Sign	29.28 ms	75,903,168	45.61 ms	118,235,215
Sign (cached SK)	23.51 ms	60,961,314	40.51 ms	105,005,378
Verify	10.40 ms	26,965,333	16.70 ms	43,293,387
Verify (cached PK)	0.59 ms	1,532,779	2.98 ms	7,737,203

5.2 Implementation details

5.2.1 Random generation

FOX makes use of a deterministic random byte generator DRBG that can generate an infinite byte string. The DRBG is implemented using a virtually infinite byte buffer, a call `DRBGGet(DRBG,n)` consumes `n` in the buffer and return them. In reality, DRBG

implements `shake256` as an XOF (eXtended Output Function), with a limited sized byte buffer. DRBG is first initialized with a contextual seed and a TAG that assures cryptographic separation with a call `DRBGInitialize(DRBG, context)` then subsequent calls to the function `shake256` are made to replenish the buffer each time it is emptied.

Based on DRBG, FOX uses a deterministic generator of elements of \mathbb{F}_q , defined by sampling and rejection.

The needed seeds are obtained by subsequent calls to the `randombytes()` function, all seeds are made 32 byte long.

Algorithm 4 Random-related Algorithms

```

1: function SAMPLEREJECT(DRBG) ▷ Return a random integer suitable for an element
   of  $\mathbb{F}_q$ 
2:   nbbits  $\leftarrow \lceil \log_2(q) \rceil$ 
3:   nbytes  $\leftarrow \lceil \log_{256}(q) \rceil$ 
4:   repeat
5:      $b \leftarrow \text{DRBG}(\text{nbytes})$  ▷  $b$  is the numeric representation (little endiand) of the
     byte string on the adequate number of bytes
6:      $b \leftarrow \text{TRUNC}(b, \text{nbbits})$  ▷  $b$  is truncated to its least significant nbbits bits
7:   until  $b < q$ 
8:   return  $b$ 
9: end function

```

5.2.2 Key generation

The secret elements S and T are linear mappings, so they can be naturally represented as matrices. Moreover, we need only to store and manipulate a sub-matrix for each of them. So for simplicity's sake, we'll keep the same notation for the matrix and its sub-block.

The key elements **Pub** and **Sec** are homogeneous quadratic polynomials, so they can be naturally represented as triangular matrices. We give here a generic procedure (see alg. 5) that generates the needed part of a matrix. The procedure `GenMatrix` shall be called for each block of matrix to be generated, with given arguments according to the needs. The benefit of this technique is that the generation of all elements can be possibly parallelized.

Algorithm 5 Generation of a block of a matrix

```

1: procedure GENMATRIX(M, seed, L, TAG, GenFunc )
2:   DRBGINITIALIZE(DRBG, TAG || seed)
3:   for  $(i, j)$  in L do
4:      $M[i, j] \leftarrow \text{GENFUNC}(\text{DRBG})$ 
5:   end for
6: end procedure

```

We define four sets of indexes (sorted in lexicographical order), and corresponding tags.

- $L_o = \{(i, j) : 1 \leq j \leq i \leq o\}$, for the Oil×Oil coefficients, TAG_O=0,
- $L_v = \{(i, j) : 1 + o \leq i \leq o + v, 1 \leq j \leq i\}$, for the Vinegar×Oil coefficients and the Vinegar×Vinegar coefficients, TAG_V=1,
- $L_S = \{(i, j) : 1 \leq i \leq t, 1 \leq j \leq o - t\}$, for **S**, TAG_S=2,
- $L_T = \{(i, j) : 1 \leq i \leq v, 1 \leq j \leq o\}$, for **T**, TAG_T=3,

We can now give a complete description of the generation algorithms used in alg. 1, see alg. 6.

Algorithm 6 Generation of the key elements

```

1: function GENERATES(seed)
2:   S  $\leftarrow$  ZEROMATRIX( $\mathbb{F}_q, t, o - t$ )
3:   GENMATRIX(S, seed,  $L_S$ , TAG_S, SampleReject)
4:   return S
5: end function

6: function GENERATE_T(seed)
7:   T  $\leftarrow$  ZEROMATRIX( $\mathbb{F}_q, v, o$ )
8:   GENMATRIX(T, seed,  $L_T$ , TAG_T, SampleReject)
9:   return T
10: end function

11: function GENERATE_PUB(seed)
12:   for  $i \leftarrow 1, o$  do
13:     Pub $_i$   $\leftarrow$  ZEROMATRIX( $\mathbb{F}_q, o + v, o + v$ )
14:     GENMATRIX(Pub $_i$ , seed,  $L_v$ , TAG_V|| $i$ , SampleReject)
15:   end for
16:   for  $i \leftarrow 1, t$  do
17:     GENMATRIX(Pub $_i$ , seed,  $L_o$ , TAG_0|| $i$ , SampleReject)
18:   end for
19:   return Pub
20: end function

```

Algorithm 7 Solving Algorithms

```

1: function SOLVELINEAR( $L, o, t$ )
     $\triangleright L$  is an affine system of  $o - t$  equations in  $o$  variables
2:    $R \leftarrow \text{ROWECHOLONFORM}(L)$ 
3:   if  $\text{RANK}(R) < o - t$  then return false,  $\perp$ 
4:   end if
5:   NP  $\leftarrow$  list of positions of non pivots  $\triangleright$  free variables
6:   P  $\leftarrow$  list of positions of pivots  $\triangleright$  dependant variables can be expressed as affine
    combinations of the free variables
7:    $H \leftarrow$  affine expressions (i.e. generic solution in free variables)  $\triangleright H$  is a
     $o \times (t + 1)$ -matrix (includes constant terms)
8:   return true,  $H$ 
9: end function

10: function SOLVEQUADRATIC( $Q, \text{hint}$ )
     $\triangleright Q$  is a quadratic system of  $t$  equations in  $t$  variables
11:    $G_1 \leftarrow \text{GROEBNERBASIS}(Q, \text{"grevlex"})$ 
12:   if  $G_1$  is not regular then return false,  $\perp$ 
13:   end if
14:    $G_2 \leftarrow \text{GROEBNERBASIS}(G_1, \text{"lex"})$   $\triangleright$  Change monomial order from "grevlex" to "lex"
15:   if  $G_2$  is not regular then return false,  $\perp$ 
16:   end if
17:   HasSolution,  $a_t \leftarrow \text{SOLVEUNIVARIATE}(G_2[t], \text{hint})$   $\triangleright$  The last polynomial of a
    regular "lex" Gröbner Basis must be univariate in  $x_t$ 
18:   if not HasSolution then return false,  $\perp$ 
19:   end if
20:   for  $i \leftarrow 1, t - 1$  do
21:      $a_i \leftarrow -\text{EVALUATE}(G_2[i] - x_i, a_t)$   $\triangleright$  The  $i^{\text{th}}$  polynomial of a regular "lex"
    Gröbner Basis must be of the kind  $x_i +$  a univariate polynomial in  $x_t$ 
22:   end for
23:   return true,  $a_1, \dots, a_t$ 
24: end function

```

5.2.3 Hash

The FOX signing process requires a Hash function that randomly sends arbitrary long messages into the vector space $(\mathbb{F}_q)^o$. This can be easily achieved by alg. 8. SeedPub is used as a publicly known constant diversifier available to both signer and verifier. Secondly, we take benefit of this function to generate a seed used for the generation of the vinegar. For the use of DRBG, TAG_M is defined as 4.

Algorithm 8 Hash of a Message and seed for signing

```

1: function HASHANDSEED(Message)
2:   Hash  $\leftarrow \text{ZEROVECTOR}(\mathbb{F}_q, o)$ 
3:   DRBGINITIALIZE(DRBG, TAG_M || SeedPub || Message)
4:   for  $i \leftarrow 1, o$  do
5:     Hash[ $i$ ]  $\leftarrow \text{SAMPLEREJECT}(\text{DRBG})$ 
6:   end for
7:   SeedHash  $\leftarrow \text{DRBG}(32)$ 
8:   return Hash, SeedHash
9: end function

```

5.2.4 Vinegar generation

The FOX signing process requires the generation of random values for the Vinegar variables, plus one value for a hint. This generation can occur several times during the signing process, hence the use of a counter. A seed is produced from the message and the secret seed, so that an attacker may not guess this process. We detail the generation of this seed and the vinegar in alg. 9. For the use of DRBG, `TAG_Vinegar` is defined as 5

Algorithm 9 Vinegar Generation

```

1: function GENERATEVINEGAR(SeedVinegar,counter)
2:    $V \leftarrow \text{ZEROVECTOR}(\mathbb{F}_q, o)$ 
3:   DRBGINITIALIZE(DRBG, TAG_Vinegar || SeedVinegar || counter)
4:   for  $i \leftarrow 1, v$  do
5:      $V[i] \leftarrow \text{SAMPLEREJECT}(\text{DRBG})$ 
6:   end for
7:   hint  $\leftarrow \text{SAMPLEREJECT}(\text{DRBG})$ 
8:   return V, hint
9: end function

```

6 Expected strength (2.B.4) for each parameter set

The security of FOX has been determined under key recovery attack and forgery attack. In the first case, the adversary tries to recover the secret key given the public key. In the forgery attack, the adversary \mathcal{A} tries to win the EUF-CMA (Existential UnForgeability under Chosen-Message Attack). To this end, \mathcal{A} interacts with the signing oracle for message chosen by \mathcal{A} , and at the end, he must output a non-already signed message along with a valid signature. It is important to note that the key recovery attack is the best attack on UOV schemes. Indeed, we are not aware of any attack exploiting the interaction with the signing oracle, better than recovering the secret oil subspace.

Consequently, we consider the extended Kipnis-Shamir attack presented in [KPG99], the generalization by Beullens in the intersection attack [Beu21], and the direct attack by solving a quadratic system. To apply these attacks on FOX, we need to cancel the $\hat{\tau}$ technique on two or more quadratic forms. To cancel the t random forms for one form, we need to compute on average q^t linear combinations. If we want more forms, the complexity becomes q^{2t} or q^{3t} times the complexity of these attacks. The FOX parameters have been determined by examining the complexity of the attacks in the small field. If we solve the equations in the extension field, the number of variables is reduced, but the degree of the equations becomes higher and the complexity will be higher.

- **FOX-I.** The complexity of the best attack is given by the HybridXLWiedemann algorithm with time complexity $2^{144.6}$ and memory complexity $2^{68.6}$ according to the MQEstimator tool. The complexity of the Kipnis-Shamir attack is 2^{207} in time but since we need to cancel the t random forms with complexity q^{2t} since we need at least 2 such forms, the total complexity becomes 2^{334} . The intersection attack requires finding at least 3 equations without errors, its complexity is thus $\geq q^{3t} \approx 2^{191}$.
- **FOX-III.** For FOX-III, the complexity of the best attack is HybridF5 in time $2^{209.5}$ and memory $2^{182.0}$. The complexity of Kipnis-Shamir is 2^{432} and with at least two equations, the complexity of this attack becomes $q^{2t} \times 2^{432} = 2^{624}$. The intersection attack requires finding at least 3 equations without errors, its complexity is thus $\geq q^{3t} \approx 2^{288}$.

Algorithm	\log_2 Time Complexity	\log_2 Memory Complexity
HybridF5	145.5	115.8
Crossbread	147.1	105
HybridXLWiedemann	144.6	68.6

Figure 2: Complexity of the Algorithms against FOX-I.

Algorithm	\log_2 Time Complexity	\log_2 Memory Complexity
HybridF5	209.5	182.0
Crossbread	287.93	97.4
HybridXLWiedemann	216.33	102.8

Figure 3: Complexity of the Algorithms against FOX-III.

- **FOX-V.** For FOX-V, the complexity of the best attack is HybridF5 in time $2^{294.5}$ and $2^{246.0}$ in memory. The complexity of Kipnis-Shamir is 2^{784} and so the overall complexity is $q^{2t} \times 2^{784} = 2^{1040}$. The intersection attack requires finding at least 3 equations without errors, its complexity is thus $\geq q^{3t} \approx 2^{384}$.

Algorithm	\log_2 Time Complexity	\log_2 Memory Complexity
HybridF5	294.7	246.0
Crossbread	597.5	106.3
HybridXLWiedemann	301.9	135.8

Figure 4: Complexity of the Algorithms against FOX-V.

In conclusion, we summarize the security level for the 3 variants of FOX.

Variant	Required Security	Estimated Security	Best Attack
FOX-I	143	144.6	HybridXLWiedemann
FOX-III	207	209.5	HybridF5
FOX-V	272	294.7	HybridF5

Figure 5: Parameter sets and corresponding key and signature sizes for the FOX signature scheme.

- **FOX is EUF-CMA attack.** We do not have a security proof for FOX. There exists a security proof for UOV in [SSH11, KX22] in the ROM and QROM. It is however much harder to adapt these proofs to FOX as we have to take into account quadratic forms. The security assumption for FOX is weaker than for UOV as there is no common isotropic subspace in FOX, contrary to UOV.
- **The BUFF attack.** In order to avoid the BUFF attack [CDF⁺21], we hash the public key and the message. Since the public key is large, we prefer to hash the public key first and then to hash the message with the hash of the public key when signing. The consequence, is that when we verify the signature, we also have to verify the hash of the public key.

7 Analysis of known attacks (2.B.5)

In this section, we will analyse the resistance of FOX against all known attacks on UOV [FIKT21].

7.1 Direct signature forgery attacks

In the forgery attack, the attacker tries to invert the quadratic map $\mathbf{Pub}(x) = h$. So we estimate its complexity as solving a random quadratic system of o equations in $n = (o + v)$ variables over \mathbb{F}_q . We use the hybrid attack by fixing k variables in [BFP09].

$$\min_k \left(3q^k \binom{n-k+d_{\text{reg}}}{d_{\text{reg}}}^2 \binom{n-k}{2} \right)$$

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

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

is non-positive. We do not consider the polynomial terms in the estimate to lower bound the complexity. To estimate this attack we also use the tool given by Bellini *et al.* in [BMSV22] with many other quadratic system solvers.

7.2 Kipnis-Shamir attack

In the Kipnis-Shamir attack [KS98], the attacker tries to recover directly the oil space by combining two quadratic forms. The attack works when the number of variables is twice the number of equations: $o + v = 2o$, *i.e.* $v = o$. It has been extended when $o + v \geq 2o$, *i.e.* when $v > o$, to q^{v-o} times polynomial factors in [KPG99], and we remove the polynomial factors. In order to recover at least two equations without the FOX transformation, we need to linearly combine on average q^{2t} equations to cancel the t full quadratic equations. Consequently, we estimate the complexity of this attack to $q^{2t} \times q^{v-o}$, that is q^{3t} .

7.3 Intersection attack

This attack tries to recover k vectors in the intersection of $\cap_{i=1}^k M_i O$, for some matrices M_i and O describes the Oil space. The attack works when the intersection is not empty, which means when $v + o < \frac{2k-1}{k-1}o$, *i.e.* when $v < \frac{k}{k-1}o$ [Beu21]. The cost of the attack boils down to solving a system of a random system of $\binom{k+1}{2}o - 2\binom{k}{2}$ equations in $k(v+o) - (2k-1)o$ variables. We must estimate the parameters k and we choose the same strategy as in [BCH⁺23]. To evaluate the cost of this attack on FOX parameters, we use the estimator of Bellini *et al.* in [BMSV22] that we adapted to consider these attacks on UOV and we take into account the number of forms we need to mount these attacks.

7.4 Quantum analysis

We take into account the quantum version of the direct attack as specified in [SW16, FHK⁺17] for solving quadratic systems with $m = n$ equations in n variables over \mathbb{F}_2 in time $O(2^{0.462n})$ quantum gates. For quantum attacks, we can also consider a Groverized version of the HybridF5 with complexity:

$$\min_k \left(3q^{k/2} \binom{n-k+d_{\text{reg}}}{d_{\text{reg}}}^2 \binom{n-k}{2} \right)$$

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

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

is non-positive.

7.5 The Rectangular Minrank Attack

This attack (described in [Beu20a]) exploits the fact that a specific (rectangular) matrix built up by composing the polar forms of the public polynomials by a random vector may have a low rank if the vector happens to belong to the secret Oil space. Therefore solving a Minrank problem enables to find the secret Oil space. Using the notations of [Beu20a], the rectangular matrix can be expressed as the representation of the linear mapping L_x defined by:

$$L_x(y) = (\Delta\mathbf{Pub}(x, y))$$

where $\Delta\mathbf{Pub}(x, y) = \mathbf{Pub}(x + y) - \mathbf{Pub}(x) - \mathbf{Pub}(y)$ is the polar form of the public key.

In the case of FOX, this attack does not apply. Possibly, the extension of this attack (a padded rectangular Minrank attack) could apply, but for the parameters of FOX, it is highly inefficient.

8 Advantages and limitations (2.B.6)

The advantages of FOX are the following.

- **Short signatures.** FOX signature sizes are respectively 124, 186, and 462 bytes for the NIST security levels I, III, and V. This is much shorter than currently selected signature schemes for Falcon (666 bytes for NIST Level I), Dilithium (2,490 bytes for NIST Level I), and Sphincs+ (over 8,000 bytes for NIST Level I). This can be compared with ECDSA of 64 bytes for NIST Level I.
- **Fast verification.** The verification just performs some hashing, and evaluates the public key form, which is very fast.
- **Simplicity.** The principle is easy to understand, since it is a hash-and-sign signature.
- **Public key of reasonable size.** The public key sizes are 50.3 Kbytes, 183 Kbytes and 695 Kbytes respectively for the different security levels. This is rather uncommon in multivariate cryptography, where for the NIST Level I parameters of UOV the public key size is about 48 Kbytes. MAYO is an exception in the MQ family as it has small public key of the same magnitude as Falcon, 900 bytes. It is worth remembering the size of Dilithium public key is 1,312 bytes.
- **UOV-based.** UOV is a well-known scheme studied for more than 20 years.
- **Strong security arguments.** We think the $\hat{+}$ technique increases the security of FOX since it can be viewed as an intermediate scheme between UOV and a full random system of quadratic equations.

The limitations are the following.

- **Large public keys.** Despite all the techniques used to compress the public key, the main limitation of FOX is its size, which is large compared to elliptic curve, isogeny-based, structured lattice-based, hash-based, and MPC-in-the-head signature schemes. The size of FOX-I public key is 56 times larger than Falcon and 38 times Dilithium, which could be problematic in some applications.

- **Signature process a little more complex than UOV.** Contrary to UOV, we need to solve a small system of quadratic equations in a small number of variables. The literature in this area is rather small, but we think it can be a good idea to solve this new problem efficiently.

In conclusion, FOX has attractive properties and performances that match the new NIST call for proposals: diversity in the assumptions, very short signature size, reasonable size public key, and strong security arguments. We think there is room of improvements for the implementations.

References

- [ACK⁺23] Thomas Aulbach, Fabio Campos, Juliane Krämer, Simona Samardjiska, and Marc Stöttinger. Separating oil and vinegar with a single trace. *Cryptology ePrint Archive*, Report 2023/335, 2023. <https://eprint.iacr.org/2023/335>.
- [BCH⁺23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *Cryptology ePrint Archive*, Report 2023/059, 2023. <https://eprint.iacr.org/2023/059>.
- [Beu20a] Ward Beullens. Improved cryptanalysis of uov and rainbow. *Cryptology ePrint Archive*, Paper 2020/1343, 2020. <https://eprint.iacr.org/2020/1343>.
- [Beu20b] 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, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.
- [Beu21] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. *LNCS*, pages 348–373. Springer, Heidelberg, Germany, 2021.
- [Beu22a] Ward Beullens. Breaking rainbow takes a weekend on a laptop. *LNCS*, pages 464–479, Santa Barbara, CA, USA, 2022. Springer, Heidelberg, Germany.
- [Beu22b] Ward Beullens. MAYO: Practical post-quantum signatures from oil-and-vinegar maps. *LNCS*, pages 355–376. Springer, Heidelberg, Germany, 2022.
- [BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.*, 3(3):177–197, 2009.
- [BMSV22] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier A. Verbel. An estimator for the hardness of the MQ problem. *LNCS*, pages 323–347, 2022.
- [BP17] Ward Beullens and Bart Preneel. Field lifting for smaller UOV public keys. In Arpita Patra and Nigel P. Smart, editors, *INDOCRYPT 2017*, volume 10698 of *LNCS*, pages 227–246, Chennai, India, December 10–13, 2017. Springer, Heidelberg, Germany.
- [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, Hyderabad, India, December 12–15, 2010. Springer, Heidelberg, Germany.

-
- [CDF⁺21] Cas Cremers, Samed Düzlül, Rune Fiedler, Marc Fischlin, and Christian Janson. BUFFing signature schemes beyond unforgeability and the case of post-quantum signatures. pages 1696–1714. IEEE Computer Society Press, 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, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.
- [CHR⁺20] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. MQDSS specifications. 2020. Version 2.1, <https://mqdss.org/index.html>.
- [DDS⁺20] Jintai Ding, Joshua Deaton, Kurt Schmidt, Vishakha, and Zheng Zhang. Cryptanalysis of the lifted unbalanced oil vinegar signature scheme. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 279–298, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.
- [DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 164–175, New York, NY, USA, June 7–10, 2005. Springer, Heidelberg, Germany.
- [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.
- [FI23] Hiroki Furue and Yasuhiko Ikematsu. A new security analysis against MAYO and QR-UOV using rectangular minrank attack. In Junji Shikata and Hiroki Kuzuno, editors, *Advances in Information and Computer Security - 18th International Workshop on Security, IWSEC 2023, Yokohama, Japan, August 29-31, 2023, Proceedings*, volume 14128 of *Lecture Notes in Computer Science*, pages 101–116. Springer, 2023.
- [FIKT21] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. A new variant of unbalanced Oil and Vinegar using quotient ring: QR-UOV. *LNCS*, pages 187–217. Springer, Heidelberg, Germany, 2021.
- [FKNT22] Hiroki Furue, Yutaro Kiyomura, Tatsuya Nagasawa, and Tsuyoshi Takagi. A new fault attack on uov multivariate signature scheme. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography*, pages 124–143, Cham, 2022. Springer International Publishing.
- [FmRPP22] Jean-Charles Faugère, Gilles macario Rat, Jacques Patarin, and Ludovic Perret. A new perturbation for multivariate public key schemes such as HFE and UOV. Cryptology ePrint Archive, Report 2022/203, 2022. <https://eprint.iacr.org/2022/203>.
- [GD24] Hao Guo and Jintai Ding. A practical minrank attack against VOX. *IACR Cryptol. ePrint Arch.*, page 166, 2024.
- [INT23] Yasuhiko Ikematsu, Shuhei Nakamura, and Tsuyoshi Takagi. Recent progress in the security evaluation of multivariate public-key cryptography. *IET Inf. Secur.*, 17(2):210–226, 2023.

- [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, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.
- [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, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.
- [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>.
- [Pat97] Jacques Patarin. The oil and vinegar signature scheme. Dagstuhl Workshop on cryptography, September, 1997.
- [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, Hyderabad, India, December 12–15, 2010. Springer, Heidelberg, Germany.
- [Péb24a] Pierre Pébereau. Singular points of UOV and VOX. *IACR Cryptol. ePrint Arch.*, page 219, 2024.
- [Péb24b] Pierre Pébereau. Subfield attack: leveraging composite-degree extensions in the quotient ring transform. *IACR Cryptol. ePrint Arch.*, page 196, 2024.
- [PTBW11] Albrecht Petzoldt, Enrico Thomae, Stanislav Bulygin, and Christopher Wolf. Small public keys and fast verification for Multivariate Quadratic public key systems. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 475–490, Nara, Japan, September 28 – October 1, 2011. Springer, Heidelberg, Germany.
- [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, Tapei, Taiwan, November 29 – December 2 2011. Springer, Heidelberg, Germany.
- [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.
- [TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Efficient key recovery for all HFE signature variants. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 70–93. Springer, 2021.