Vox
Specification v1.0 – 06/01/2023

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

VOX is a UOV-based hash-and-sign signature scheme from the Multivariate Quadratic (MQ) family. In order to reduce the size of the public key, we use a new variant proposed by Faugère, Macario-Rat, Patarin, and Perret under the name UOV$\hat{+}$ and the QR (Quotient Ring) technique introduced by Furue, Ikematsu, Kiyomura, and Takagi at ASIACRYPT 2021. The 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. 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.

The size of the signature is 816 bits for VOX-I with 128 bits of security for public key size 8.9Kbytes. The verification time is very good and we can sign 1200 messages per second if the secret key is decompressed and 950 otherwise. VOX has very attractive properties, but we will also propose another tradeoff, called Full-VOX. This variant offers a double security criteria: if the security of UOV$\hat{+}$ is broken, the adversary has still to break a UOV instance. The QR technique to reduce the size of the public key is also removed from this variant.

1.1 General introduction

Multivariate Cryptography is a mature proposal in the landscape of public-key alternative 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 advantage 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 signature. 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 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,
- VOX has strong security arguments.

General Presentation of VOX. We start by quickly presenting the 4 main ideas behind the design of VOX. A detailed mathematical presentation of VOX 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 mutivariate 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 innovation of VOX 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 VOX 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 where the variables belong (we do not perform exhaustive search on t values), unlike the complexity of the attacks that seem to increase rapidly with this size.

4. **The QR technique.** This is also a relatively new technique, presented by Furue *et al.* in [FIKT21]. As with the BPB algorithm, thanks to this technique it is possible to significantly reduce the size of the public key (and we will combine the two techniques in VOX). However, unlike the BPB technique, it is not proved that the QR technique keeps the same security for the scheme.

   In VOX, with the parameters that we will choose, we are confident to use this technique, thanks to the strengthening obtained by the $\hat{+}$ technique. (However, in UOV without $\hat{+}$ we do not recommend to use it). We also propose a variant (see VOX-F below) where we will not use this technique.

VOX-F variant (VOX-Full, nickname "FOX"). VOX is our main candidate. In this document, we will present and explain the precise parameters that we will choose for VOX and the expected security for these parameters. However, we will also present a variant of VOX, called "VOX-F" (or Full VOX, nickname "FOX"). The design of VOX-F has been chosen in order to obtain easier security arguments than for VOX (but at the cost of a less efficient algorithm). Therefore, for example, in VOX-F we will not use the QR technique. Moreover if in VOX-F an attacker was able to remove all the equations introduced by the $\hat{+}$ technique, then the resulting UOV system will still have the required security against the best known attacks. Conversely, the VOX-F parameters will be chosen such that the elimination of the equation of the equations introduced by the $\hat{+}$ is a problem with at least the same complexity. Therefore the security of VOX-F is related to two problems: the elimination of the $\hat{+}$ equations and solving the resulting UOV system. (Unlike in VOX where the resulting system parameters do not have the security wanted, i.e. VOX relies on the fact that the $\hat{+}$ technique increases the security).

A short history of UOV and VOX. 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 be 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].

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 VOX 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. More precise arguments on VOX and VOX-F will be given later in this document.

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 VOX 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/VOX against side channel attacks than for most of other public key algorithms. This comes from the fact that in UOV/VOX 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 VOX 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, VOX introduces new ideas to reduce the size of the public key, which is an important parameters for bandwidth-constraint applications, which is one of the drawback of post-quantum schemes and also for MQ schemes. We think that the UOV$\hat{+}$ improves the security of UOV, which makes VOX 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. Finally, the QR transformation can be seen as a structured version of MQ schemes.

As a consequence, VOX-I achieves signature of size 102B with public key of size 9104B. The performances of key generation, signing and verification are rather good compared to other schemes. In the following, we compare VOX 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. The public key is however 10 times larger than for VOX.

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 SUSHYFISH [Beu20] 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 Vox are:

- λ, the security level of VOX, 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,
- c, a common divider of o and v, that controls the QR variant.

2.2 Secret-key and public-key

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

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

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

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

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

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

or equivalently the coefficients $a'_{i,j,k}$ are canceled for $i = t + 1, \ldots, o$ 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:

$$\mathbf{Sec} = \mathbf{S} \circ \mathbf{Pub} \circ \mathbf{T}. \tag{4}$$

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 folowing way:

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

where I is the identity matrix with given dimension, and $\mathbf{T}'$ and $\mathbf{S}'$ 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 inversible, whatever the values of $\mathbf{T}'$ and $\mathbf{S}'$.

2.2.1 The QR variant

The QR variant was originaly presented in [FIKT21]. The QR variant aims to decrease the size of public and secret key, with probably no impact on the security of the scheme. The QR variant may be enable when $c > 1$. In this case, we introduce $G = \mathbb{F}_q(g) = \mathbb{F}_q[x]/p(x)$, a field extension of $\mathbb{F}_q$ of degree c, i.e. g is a formal root of an irreducible univariate

polynomial p of degree c over $\mathbb{F}_q$. Since c divides o and v, we note $o_c = \frac{o}{c}$ and $v_c = \frac{v}{c}$. We note Φ the natural isomorphism : $(\mathbb{F}_q)^c \mapsto G$, $(x_1, \ldots, x_c) \to \sum_{i=1}^{c} g^{i-1} x_i$. We note ϕ the natural embedding : $(\mathbb{F}_q) \mapsto G, x \to x$. For a given mapping $f : A \mapsto B$, we denote the natural extension $f^{\langle n \rangle} : A^n \mapsto B^n, (x_1, \ldots, x_n) \to (f(x_1), \ldots, f(x_n))$. We introduce also the well known trace morphism $\mathrm{Tr} : \mathbb{F}_q(g) \mapsto \mathbb{F}_q, x \to \sum_{i=0}^{c-1} x^{q^i}$. Note that the QR variant may use any linear form instead of the Trace. However, since all linear forms can be expressed as $x \to \mathrm{Tr}(ax)$, for a given $a \in G$, it's just a matter of a constant choice. We use the notation $\bar{x}$ to indicate that $\bar{x}$ is in or over G. Therefore, we can define $\bar{\boldsymbol{\xi}}_i = \Phi(x_{ci+1}, \ldots, x_{ci+c})$.

In this variant, **Pub**, **Sec** and **T**, can be obtained from $\bar{\mathbf{P}}\mathbf{ub}$, $\bar{\mathbf{S}}\mathbf{ec}$ and $\bar{\mathbf{T}}$ that we present now. **S** is defined as before as a random element of $\mathrm{GL}_o(\mathbb{F}_q)$. $\bar{\mathbf{P}}\mathbf{ub}$ and $\bar{\mathbf{S}}\mathbf{ec}$ are sets of o homogeneous quadratic equations in $o_c + v_c$ variables. The o polynomials $\{\bar{p}'_1, \ldots, \bar{p}'_o\}$ of $\bar{\mathbf{S}}\mathbf{ec}$ have the generic form of homogeneous polynomials:

$$\bar{p}'_i = \sum_{1 \le k \le j \le o_c + v_c} \bar{a}_{i,j,k} \bar{x}_j \bar{x}_k \tag{6}$$

where $\bar{a}'_{i,j,k}$ are elements of G. Accordingly, the last $o - t$ polynomials $\{\bar{p}'_{t+1}, \ldots, \bar{p}'_o\}$ of $\bar{\mathbf{S}}\mathbf{ec}$ have the special (UOV) form :

$$\bar{p}'_i = \sum_{\substack{1+o_c \le j \le o_c+v_c \\ 1 \le k \le j}} \bar{a}_{i,j,k} \bar{x}_j \bar{x}_k. \tag{7}$$

$\bar{\mathbf{T}}$ is a random bijective linear mapping in $\mathrm{GL}_{o_c+v_c}(G)$. Moreover, $\bar{\mathbf{P}}\mathbf{ub}$, $\bar{\mathbf{S}}\mathbf{ec}$, **S** and $\bar{\mathbf{T}}$ are required to satify

$$\bar{\mathbf{S}}\mathbf{ec} = \phi^{\langle o \rangle} \circ \mathbf{S} \circ {\phi^{\langle o \rangle}}^{-1} \circ \bar{\mathbf{P}}\mathbf{ub} \circ \bar{\mathbf{T}}. \tag{8}$$

In this variant, we define then

$$T(x_1, \ldots, x_{o+v}) = {\Phi^{\langle o_c+v_c \rangle}}^{-1}(\bar{T}(\bar{\boldsymbol{\xi}}_1, \ldots, \bar{\boldsymbol{\xi}}_{o_c+v_c})) \tag{9}$$

and

$$\begin{aligned} p_i(x_1, \ldots, x_{o+v}) &= \mathrm{Tr}(\bar{p}_i(\bar{\boldsymbol{\xi}}_1, \ldots, \bar{\boldsymbol{\xi}}_{o_c+v_c})) \quad i = 1, \ldots, o, \\ p'_i(x_1, \ldots, x_{o+v}) &= \mathrm{Tr}(\bar{p}'_i(\bar{\boldsymbol{\xi}}_1, \ldots, \bar{\boldsymbol{\xi}}_{o_c+v_c})) \quad i = 1, \ldots, o, \end{aligned} \tag{10}$$

In this variant, as soon as eqs. (5), (6), (7), (8), and (9) are satisfied, then eqs. (1), (2), (3) and (4) are also satisfied.

2.2.2 Matrix expression of the QR variant

The QR variant is designed so the matrix expressions of $\mathbf{T}, \mathbf{Pub}$ and **Sec** can easyly be deduced from those of $\bar{\mathbf{P}}\mathbf{ub}$, $\bar{\mathbf{S}}\mathbf{ec}$, and $\bar{\mathbf{T}}$. Each coefficient in $\bar{\mathbf{P}}\mathbf{ub}$, $\bar{\mathbf{S}}\mathbf{ec}$ corresponds to a $c \times c$-block in **Pub** and **Sec**: for a coefficient $\bar{a}$, we get a block $W_{\bar{a}}$ whose coefficient i, j is $\mathrm{Tr}(\bar{a} g^{i+j-2})$. With this convention, one can easyly check that :

$$(x_{ci+1}, \ldots, x_{ci+c}) {W_{\bar{a}}}^t (x_{cj+1}, \ldots, x_{cj+c}) = \mathrm{Tr}(\bar{a} \bar{\xi}_i \bar{\xi}_j)$$

Each coefficient of $\bar{\mathbf{T}}$ corresponds to a $c \times c$-block in $\mathbf{T}$: for a coefficient $\bar{a}$, we get a block $U_{\bar{a}}$ whose row i is $\Phi^{-1}(\bar{a} g^{i-1})$. With this convention, one can easyly check that :

$$(x_{ci+1}, \ldots, x_{ci+c}) U_{\bar{a}} = \Phi^{-1}(\bar{a} \bar{\xi}_i)$$

Furthermore this block notation enables to notice that, for all $a_1, a_2, a_3 \in G$

$$U_{a_1} + U_{a_2} = U_{a_1+a_2}$$

$$W_{a_1} + W_{a_2} = W_{a_1+a_2}$$
$$U_{a_1} W_{a_2} {}^t U_{a_3} = W_{a_1 a_2 a_3}$$

Remark 1. For implementation's sake, we can start by computing U_g as the companion matrix of the polynomial p, and W_1 as the matrix whose i,j-coefficient is $\mathrm{Tr}(g^{i+j-2})$. The other matrices are obtained by the following formulas :

$$U_{\Phi(a_1,\ldots,a_c)} = \sum_{i=1}^{c} a_i U_g^{i-1}$$

and

$$W_{\Phi(a_1,\ldots,a_c)} = \sum_{i=1}^{c} a_i U_g^{i-1} W_1$$

Remark 2. The matrix companion of the monic univariate polynomial of degree c, $x^c + a_{c-1}x^{c-1} + \cdots + a_1 x + a_0$ is by definition the $c \times c$ matrix

$$\begin{pmatrix} 0 & 1 & 0 & \ldots & & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & & 1 & 0 \\ 0 & & \cdots & & 0 & 1 \\ -a_0 & -a_1 & & \cdots & & -a_{c-1} \end{pmatrix}.$$

Remark 3. Each $c \times c$-block in $\mathbf{Pub}, \mathbf{Sec}$ and $\mathbf{T}$, has c^2 coefficients. However due to the basis representation given above, they can be stored as c coefficients only, hence the gain of size.

Remark 4. As also pointed out in the original paper [FIKT21], the choice of even q should be avoided. The reason is that in this case, the quadratic forms associated with the matrices W_a set in the diagonal blocks of public and secret keys, are diagonal forms. Therefore, the choice of q even would lead to weak keys.

2.2.3 The BPB Compression

This feature has been introduced in [PBB10, BPB10]. Its purpose is also to decrease the size of the public key. It can be used in conjunction with the QR variant or without it and 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:

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

2.2.4 Key generation process

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

Algorithm 1 Key Generation for Compressed VOX public keys

procedure CryptoSignKeyPair(sk, pk)
  SeedSec ← `randombytes(32)`
  $\mathbf{S} \leftarrow$ GenerateS(SeedSec)
  $\bar{\mathbf{T}} \leftarrow$ GenerateT(SeedSec)

  SeedPub ← `randombytes(32)`
  $\bar{\mathbf{Pub}}_v \leftarrow$ GeneratePub(SeedPub)
  $\bar{\mathbf{Pub}}_o \leftarrow -(\mathbf{S} \circ \bar{\mathbf{Pub}}_v \circ \bar{\mathbf{T}})_o$

  $\bar{\mathbf{Sec}} \leftarrow \mathbf{S} \circ \bar{\mathbf{Pub}} \circ \bar{\mathbf{T}}$
  sk ← encodeSK(SeedSec, SeedPub, $\bar{\mathbf{Sec}}$)
  pk ← encodePK(SeedPub, $\bar{\mathbf{Pub}}_o$)
end procedure

2.2.5 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 $32 + \lceil((o-t)*o*(o/c+1)/2*\lceil\log_2(q)\rceil)/8\rceil$ bytes.

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

In VOX, q has been choosen 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

Vox uses the Hash and Sign paradigm. To sign a message M, Vox 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, Vox 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. 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 variables [1]. 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

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

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.

Algorithm 2 Signature Algorithm

function VoxSign(hash, SeedVinegar, $\mathbf{S}, \mathbf{T}, \mathbf{Sec}$)
  $y \leftarrow \text{hash} \times \mathbf{S}$
  count $\leftarrow 0$
  repeat
    repeat
      vinegar,hint $\leftarrow$ GenerateVinegar(SeedVinegar,count) ▷ Deterministic random vinegar + milestone for choosing solution
      count $\leftarrow$ count $+1$
      $L \leftarrow$ Evaluate($\mathbf{Sec}^o - y_o$, vinegar)
      success,$H \leftarrow$ SolveLinear(L)
    until success ▷ here, L is regular
    $Q \leftarrow$ Evaluate($\mathbf{Sec}^v - y_v$, vinegar, H)
    success,$\xi \leftarrow$ SolveQuadratic(Q, hint)
  until success ▷ here, Q is regular and has a solution
  oil $\leftarrow H \times \xi$
  $z \leftarrow$ vinegar $\|$ oil
  $\sigma \leftarrow z \times \mathbf{T}$
  return σ
end function

procedure CryptoSign(SignedMessage, Message, sk)
  SeedSec, SeedPub, $\mathbf{T}, \mathbf{S}, \mathbf{Sec} \leftarrow$ decodeSK(sk)
  hash, SeedVinegar $\leftarrow$ HashAndSeed(Message, SeedPub)
  $\sigma \leftarrow$ VoxSign(hash, SeedVinegar $\|$ SeedSec, $\mathbf{S}, \mathbf{T}, \mathbf{Sec}$)
  SignedMessage $\leftarrow$ EncodeSignature(Message, σ)
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.

Remark 5. One can also compute instead $\mathrm{Tr}(\bar{\mathbf{P}}\mathbf{ub}(\Phi^{\langle o_c+v_c\rangle}(\sigma)))$, since this leads to the same result.

Algorithm 3 Verification Algorithm

function VoxVerify(σ, hash, **Pub**)
    Result $\leftarrow$ true
    for $i \leftarrow 1, o$ **do**
        Result $\leftarrow$ Result AND (σ_i EQUAL Evaluate(p_i,hash))
    end for
    return Result
end function

function CryptoSignOpen(SignedMessage, Message, pk)
    SeedPub, **Pub** $\leftarrow$ decodePK(pk)
    Message, $\sigma \leftarrow$ decodeSignedMessage(SignedMessage)
    hash, $\perp \leftarrow$ HashAndSeed(Message, SeedPub)
    return VoxVerify(σ, hash, **Pub**)
end function

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

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

Variant	Security Level	q	o	v	t	c	|sig|	|cpk|	|sk|	|csk|
VOX-I	128	251	8	9	6	6	102 B	9,104 B	35,056 B	64 B
VOX-III	192	1021	10	11	7	7	184 B	30,351 B	111,297 B	64 B
VOX-V	256	4093	12	13	8	8	300 B	82,400 B	292,160 B	64 B

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

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

VOX 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 $t > 0$, VOX 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 **Pub**, find O, a o-dimensional subspace such that **Pub** vanishes on O.*

becomes for VOX:

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

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

Regarding the inversion of the trapdoor of VOX, 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 VOX, is that any attack against it would primarily be a UOV-attack, where the attacker selects at random equations in the VOX public key-space, and hopes that by chance that they are in fact UOV-like. Indeed, if one selects a random linear combinaison of VOX 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, we believe that any distinguishing key attack against UOV requires at least 3 equations. Therefore choosing the parameters such as $q^{3t} > 2^\lambda$ prevents any UOV-like attack.

In the same vein, we have also imagined to base VOX on the original (balanced) Oil and Vinegar. In this particular case, we know that two equations of OV can be distinguished from two random ones. Therefore we think that parameters satisfying $c = v = o$ and $q^{2t} > 2^\lambda$ may lead to interesting choices whith significantly shorter public keys and comparable signature size. These cases need some further investigations.

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 publick key, since it helps to represent them in a condensed way. It is obviously also the case for the VOX scheme.

In the same vein, the QR modification (with odd q) seems to have no impact on the security, since apparently the modelisation of any attack on VOX seems easier in the ground field than in the extension field, therefore any information about the QR structure is lost by the attacker.

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.

- **VOX-I Benchmarks: average of 1000 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	0.39 ms	707,796	1.30 ms	2,350,064
Sign	0.37 ms	664,265	0.38 ms	679,237
Sign (cached SK)	0.27 ms	488,308	0.29 ms	517,285
Verify	0.09 ms	168,567	0.22 ms	396,311
Verify (cached PK)	0.02ms	44,085	0.07 ms	134,240

- **VOX-III Benchmarks: average of 1000 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	2.22 ms	4,006,402	5.17 ms	9,334,246
Sign	1.50 ms	2,709,851	1.63 ms	2,937,318
Sign (cached SK)	1.05 ms	1,887,598	1.14 ms	2,061,593
Verify	0.40 ms	713,968	0.86 ms	1,548,342
Verify (cached PK)	0.08 ms	141,778	0.27 ms	486,889

- **VOX-V Benchmarks: average of 1000 iterations**

	Timing (AVX2)	Number of cycles (AVX2)	Timing (C)	Number of Cycles (C)
Keygen	7.14 ms	12,893,376	14.46 ms	26,097,363
Sign	6.71 ms	12,110,394	7.33 ms	13,226,130
Sign (cached SK)	4.93 ms	8,902,607	5.41 ms	9,772,441
Verify	0.88 ms	1,585,504	2.07 ms	3,734,592
Verify (cached PK)	0.14 ms	259,305	0.65 ms	1,180,465

5.2 Implementation details

In the implementation, we use the following definition for the field extensions.

	VOX-I	VOX-III	VOX-V
Polynomial	$X^6 + X + 1$	$X^7 + X + 5$	$X^8 + 2$

5.2.1 Random generation

In alg. 4 can be found the tools used for managing the conversion between the extension field $\mathbb{F}_{q^c}$ and the ground field $\mathbb{F}_q$.

Algorithm 4 QR-related Algorithms

procedure AlgebraicSettings(q, c)
  $p \leftarrow_\$$ an irreducible polynomial of degree c over $\mathbb{F}_q$
  $g \leftarrow_\$$ a root of p
  $U \leftarrow$ MatrixCompanion(p)
  $W \leftarrow$ ZeroMatrix($\mathbb{F}_q, c, c$)
  for $(i, j) \leftarrow (1, 1), (c, c)$ **do**
    $W[i, j] \leftarrow \mathrm{Tr}(g^{i+j-2})$
  end for
  $I \leftarrow$ IdentityMatrix($\mathbb{F}_q, c$)
  return U, W, I
end procedure

function CoefficientExpansion($\bar{\xi}$, U, W, c)
  return $\sum_{i=1}^{c} \xi_i U^{i-1} W$
end function

function MatrixExpansion($\bar{M}$, U, W, c)
  $n' \leftarrow$ Dimension($\bar{M}$)
  $n \leftarrow n' \times c$
  $M \leftarrow$ ZeroMatrix($\mathbb{F}_q, n, n$)
  for $(i, j) \leftarrow (1, 1), (n', n')$ **do**
    M-c-block$[i, j] \leftarrow$ CoefficientExpansion($\bar{M}[i, j], U, W, c$)
  end for
  return M
end function

VOX 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 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, then subsequent calls to the function `shake256` are made to replenish the buffer each time it is emptied.

Based on DRBG, VOX uses a deterministic generator of elements of $\mathbb{F}_q$, defined by sampling and rejection. An immediate version for the field extension $\mathbb{F}_{q^c}$ is defined. See alg 5.

The random generation of an element of $\bar{a} \in \mathbb{F}_{q^c}$ is performed by c subsequent calls to SampleRejection to generate its coordinates $(a_0, \ldots, a_{c-1})$ in this order.

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

Algorithm 5 Random-related Algorithms

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

function SAMPLEREJECTEXT(DRBG)
    for $i \leftarrow 1, c$ **do**
        $a_i \leftarrow$ SAMPLEREJECT(DRBG)
    end for
    return $a_1, \ldots, a_c$
end function

5.2.2 Key generation

The secret elements S, T and $\bar{T}$ 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**, **Sec**, $\bar{\textbf{Pub}}$, and $\bar{\textbf{Sec}}$, are homogeneous quadratic polynomials, so they can be naturally represented as triangular matrices. We give here a generic procedure (see alg. 6) 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 6 Generation of a block of a matrix

procedure GENMATRIX(M,seed,L, TAG, GenFunc)
    INITIALIZE(DRBG, seed ‖ TAG)
    for (i, j) in L **do**
        M$[i, j] \leftarrow$ GENFUNC(DRBG)
    end for
end procedure

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

- $L_o = \{(i, j) : 1 \leq j \leq i \leq o_c\}$, for the Oil×Oil coefficients, TAG_O=0,
- $L_v = \{(i, j) : 1 + o_c \leq i \leq o_c + v_c, 1 \leq j \leq i\}$, 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,
- $L_T = \{(i, j) : 1 \leq i \leq v_c, 1 \leq j \leq o_c\}$, for $\bar{\textbf{T}}$, TAG_T=3,

We can now give a complete description of the generation algorithms used in alg. 1, see alg. 7.

Algorithm 7 Generation of the key elements

function GenerateS(`seed`)
  $\mathbf{S} \leftarrow$ ZeroMatrix($\mathbb{F}_q$, $t, o-t$)
  GenMatrix($\mathbf{S}$, `seed`, $L_{\mathbf{S}}$, TAG_$\mathbf{S}$, SampleReject)
  return S
end function

function GenerateT(`seed`)
  $\bar{\mathbf{T}} \leftarrow$ ZeroMatrix($\mathbb{F}_{q^c}$, v_c, o_c)
  GenMatrix($\bar{\mathbf{T}}$, `seed`, $L_{\mathbf{T}}$, TAG_$\mathbf{T}$, SampleRejectExt)
  return $\bar{\mathbf{T}}$
end function

function GeneratePub(`seed`)
  for $i \leftarrow 1, o$ **do**
    $\bar{\mathbf{Pub}}_i \leftarrow$ ZeroMatrix($\mathbb{F}_{q^c}$, $o_c + v_c, o_c + v_c$)
    GenMatrix($\bar{\mathbf{Pub}}_i$, `seed`, L_v, TAG_V$\|i$, SampleRejectExt)
  end for
  for $i \leftarrow 1, t$ **do**
    GenMatrix($\bar{\mathbf{Pub}}_i$, `seed`, L_o, TAG_O$\|i$, SampleRejectExt)
  end for
  return $\bar{\mathbf{Pub}}$
end function

Algorithm 8 Solving Algorithms

function SolveLinear(L, o, t)
▷ L is an affine system of $o - t$ equations in o variables
  $R \leftarrow$ RowEchelelonForm(L)
  if Rank(R) $< o - t$ **then return** false, $\perp$
  end if
  NP $\leftarrow$ list of positions of non pivots ▷ free variables
  P $\leftarrow$ list of positions of pivots ▷ dependant variables can be expressed as affine combinaisons of the free variables
  $H \leftarrow$ affine expressions (i.e. generic solution in free variables) ▷ H is a $o \times (t+1)$-matrix (includes constant terms)
  return true, H
end function

function SolveQuadratic(Q, hint)
▷ Q is a quadratic system of t equations in t variables
  $G_1 \leftarrow$ GroebnerBasis(Q,"grevlex")
  if G_1 is not regular **then return** false,$\perp$
  end if
  $G_2 \leftarrow$ GroebnerBasis(G_1,"lex") ▷ Change monomial order from "grevlex" to "lex"
  if G_2 is not regular **then return** false,$\perp$
  end if
  $a_t \leftarrow$ SolveUnivariate($G_2[t]$, hint) ▷ The last polynomial of a regular "lex" Gröbner Basis must be univariate in x_t
  if no solution **then return** false,$\perp$
  end if
  for $i \leftarrow 1, t-1$ **do**
    $a_i \leftarrow -$Evaluate($G_2[i] - x_i, a_t$) ▷ The i^{th} polynomial of a regular "lex" Gröbner Basis must be of the kind x_i + an univariate polynomial in x_t
  end for
  return true, $a_1, \ldots, a_t$
end function

5.2.3 Hash

The VOX 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. 9. SeedPub is used as a publicly known constant diversifier available to both signer and verifier. Secondarily, we take benefit of this function to generate a seed used for the generation of the vinegar.

Algorithm 9 Hash of a Message and seed for signing

function HashAndSeed(Message)
  HASH $\leftarrow$ ZeroVector($\mathbb{F}_q$, o)
  Initialize(DRBG, SeedPub $\|$ Message)
  for $i \leftarrow 1, o$ **do**
    HASH[i] $\leftarrow$ SampleReject(DRBG)
  end for
  SeedVinegar $\leftarrow$ DRBG(32)
  return Hash, SeedVinegar
end function

5.2.4 Vinegar generation

The VOX 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. 10.

Algorithm 10 Vinegar Generation

```
function GenVinegar(seed,counter)
    V ← ZeroVector(𝔽_q, o)
    Initialize(DRBG, seed ‖ counter)
    for i ← 1, v do
        V[i] ← SampleReject(DRBG)
    end for
    hint ← SampleReject(DRBG)
    return V, hint
end function
```

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

The security of VOX 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. The VOX 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.

- **VOX-I.** The complexity of the best attack is given by the HybridF5 algorithm with time complexity 2^{134} and memory complexity 2^{111} according to the MQEvaluator tool. The complexity of the Kipnis-Shamir attack (resp. intersection attack) is 2^{61} in time (resp. 2^{45}) but since we need to cancel the t random forms with complexity q^{2t} since we need at least 2 such forms, the security becomes 2^{156} (resp. 2^{140}).

Algorithm	$\log_2$Time Complexity	$\log_2$ Memory Complexity
HybridF5	134	111
Crossbread	136.5	95
HybridXLWiedemann	140	66

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

- **VOX-III.** For VOX-III, the complexity of the best attack is HybridF5 in time $2^{197.7}$ and memory 2^{160}. The complexity of Kipnis-Shamir is 2^{84} and with at least

two equations, the complexity of this attack becomes $q^{2t} \times 2^{84} = 2^{223.9}$. For the intersection attack, the complexity is $2^{48.8}$ with $k = 9$. In this case, we need at least 3 equations without errors, and the complexity becomes $2^{48.8} \times q^{3t} = 2^{258}$.

Algorithm	$\log_2$Time Complexity	$\log_2$ Memory Complexity
HybridF5	197.7	160.6
Crossbread	249.6	97.4
HybridXLWiedemann	204.3	91.9

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

- **VOX-V.** For VOX-V, the complexity of the best attack is 2^{271} in time and 2^{239} in memory. The complexity of Kipnis-Shamir is 2^{111} and so the overall complexity is $q^{2t} \times 2^{111} = 2^{302}$. The complexity of the intersection attack is 2^{52} with $k = 11$. Consequently, if we need at least 3 equations, the overall complexity becomes $q^{3t} \times 2^{52} = 2^{340}$.

Algorithm	$\log_2$Time Complexity	$\log_2$ Memory Complexity
HybridF5	271	239
Crossbread	443	105
HybridXLWiedemann	278	132

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

Variant	Security Level	Estimated Security	Best Attack
VOX-I	128	134	HybridF5
VOX-III	192	197.7	HybridF5
VOX-V	256	271	HybridF5

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

- **The BUFF attack.** Finally, 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 VOX against all known attacks on UOV and its variants for QR [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 VOX 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 VOX parameters, we use the estimator for

7.4 Extension of these attacks to QR variant of VOX

In the case of QR, we can either estimate the complexity of the previous attack in the small field $\mathbb{F}_q$ or in the extension field, $\mathbb{F}_{q^c}$. Solving the equations in the extension field makes the degree of the equations higher as stated in [FIKT21].

7.5 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.

8 A larger family of Full-Vox parameters

To estimate the security of VOX, the best attack consists in linearly combining the quadratic forms up to cancelling the added random forms. We need at least two such cancellations to perform the Kipnis-Shamir attack. This cost is q^{2t} and then the cost of attacking UOV, which is $q^{v-o} = q^t$ in our case. Consequently, we use the rule of thumb

$$q^{3t} \approx 2^{\lambda},$$

where λ is the security parameter. This allows us to reduce the parameters of the underlying UOV scheme. Full-VOX, a.k.a. FOX, does not consider the UOV resistance, but just the resistance to remove the UOV$\hat{+}$ transformation, and the rule of thumb is

$$q^{2t} \approx 2^{\lambda}.$$

Moreover, as the added structure of the QR is not as mature as the standard UOV, as this variant has been less studied, we remove this structure in FOX.

Parameters for expected security

Variant	Security Level	q	o	v	t	|sig|	|cpk|	|csk|
FOX-I	128	251	48	72	8	120 B	47,056 B	64 B
FOX-III	192	4093	68	106	8	261 B	211,156 B	64 B
FOX-V	256	65521	91	140	8	462 B	694,892 B	64 B

Figure 6: Parameter sets and corresponding key and signature sizes for the Full-VOX (FOX) signature scheme.

9 Advantages and limitations (2.B.6)

The advantages of VOX are the following.

- **Short signatures.** VOX signature sizes are respectively 102, 167, and 276 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 9,104; 22,082; and 63,392 bytes 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 VOX 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 VOX 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 VOX-I public key is 10 times larger than Falcon and 7 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, VOX 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-Jhih 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.

[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, 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 Düzlü, 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.

[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.

[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.

[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.

[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, 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.