SNOVA

Proposal for NISTPQC: Digital Signature Schemes project

May 25, 2023

Contents

1 Algorithm specification (2.B.1) 3

1.1 Introduction . . . 3

1.2 Preliminaries . . . 5

1.2.1 Notations and conventions . . . 5

1.2.2 Basic notions . . . 6

1.2.3 NIST security level. . . . 7

1.2.4 Unbalanced Oil and Vinegar signature (UOV) scheme . . . 7

1.3 Parameter space of the SNOVA scheme . . . 8

1.4 Design rationale . . . 9

1.5 SNOVA signature scheme . . . 9

1.5.1 Description . . . 9

1.5.2 Key generation process of SNOVA . . . 11

1.5.3 To attain EUF-CMA security . . . 12

1.6 Implementation details . . . 12

1.7 Constants and tables . . . 14

1.8 Algorithms . . . 15

1.8.1 Algorithms for key generation . . . 15
1.8.2 Algorithms for signature generation . . . 17
1.8.3 Algorithms for signature verification . . . 22
1.9 Parameters settings . . . 24
1.9.1 List of our parameters . . . 24
1.9.2 How the performance are affected by parameters . . . 25

2 Performance analysis (2.B.2) 25
2.1 Platforms used in the estimation . . . 25
2.2 Time . . . 26
2.3 Space . . . 26

3 Expected Security Strength (2.B.4) 27
3.1 Security strength . . . 27

4 Analysis With Respect To Known Attacks (2.B.5) 28
4.1 A note of our analysis . . . 28
4.2 Preliminaries . . . 30
4.2.1 Solving MQ systems and complexity estimation . . . 30
4.2.2 MinRank problem and Support-Minors modeling . . . 31
4.3 Forgery attacks . . . 32
4.3.1 Direct attack . . . 32
4.3.2 Collision attack . . . 34
4.4 Key Recovery Attacks . . . 35
4.4.1 Quadratic forms over ring and MinRank attacks . . . 35
4.4.2 Reconciliation Attack . . . 37
4.4.3 Kipnis-Shamir attack (UOV attack) . . . 37
4.4.4 Intersection attack . . . 38

4.4.5 Equivalent key attack 40

5 Advantages and limitations (2.B.6) 41

5.1 Advantages 41

5.2 Limitations 42

1 Algorithm specification (2.B.1)

In the following Introduction and Preliminaries, we borrow paragraphs and slightly modify them from Wang *et al.* [51].

1.1 Introduction

We propose a simple noncommutative-ring based UOV signature scheme with key-randomness alignment: SNOVA (or Simple NOVA, which can be viewed as a simplified version of NOVA [50]). Both NOVA and SNOVA are multivariate cryptosystems over noncommutitive rings which is indicated by the letter N. The letters O and V stand for oil and vinegar variables, respectively, as in the Unbalanced Oil and Vinegar signature (UOV) scheme. The letter A denotes that key-randomness alignment is employed.

Before NOVA [50], all known multivariate cryptosystems are systems of nonlinear polynomial equations in several variables over a finite field, say $\mathbb{F}_q$. The security of these multivariate schemes is based on the MQ problem: for m quadratic polynomials $P_1(x_1, \ldots, x_n)$, $P_2(x_1, \ldots, x_n), \ldots, P_m(x_1, \ldots, x_n)$ in n variables $x_1, x_2, \ldots, x_n$ over the chosen base finite field $\mathbb{F}_q$ of order q, to find a vector $(a_1, a_2, \ldots, a_n) \in \mathbb{F}_q^n$ such that $P_1(a_1, \ldots, a_n) = P_2(a_1, \ldots, a_n) = \cdots = P_m(a_1, \ldots, a_n) = 0$. The MQ problem is proven to be NP-hard [23]. The private key of a usual multivariate scheme consists of three maps: $S : \mathbb{F}_q^m \to \mathbb{F}_q^m$, $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$, $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ where F is a plausibly invertible polynomial map (called the central map) and S, T are easily invertible maps (usually linear maps) to hide the structure of the central map F. The public key is the composite map $S \circ F \circ T$.

Among the above multivariate schemes, by its simplicity, UOV is worth more explaining. The central map of UOV scheme $F : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is as below.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_m \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{v}\sum_{k=j}^{n} f_{1,jk}x_jx_k \\ \vdots \\ \sum_{j=1}^{v}\sum_{k=j}^{n} f_{i,jk}x_jx_k \\ \vdots \\ \sum_{j=1}^{v}\sum_{k=j}^{n} f_{m,jk}x_jx_k \end{bmatrix}$$

where $f_{i,jk}$'s are the coefficients chosen randomly from $\mathbb{F}_q$. Thus F consists of m homogeneous quadratic polynomials in n variables over $\mathbb{F}_q$ and $F_i = \vec{\mathbf{x}}^t [F_i] \vec{\mathbf{x}}$ with $\vec{\mathbf{x}} = (x_1, \cdots, x_n)^t$. Note that, for $j, k = v+1, \cdots, n$, each F_i does not contain x_jx_k terms. This kind of phenomenon is analogous to that oil and vinegar won't mix completely and this enables us to invert F easily.

The variables $x_1, \cdots, x_v$ are called the vinegar variables, and $x_{v+1}, \cdots, x_n$ oil variables. It is required that $v > o$ in order to resist the K-S attack[28] on the OV scheme[36]. This is the reason why the scheme is called Unbalanced Oil and Vinegar (UOV).

The design of UOV chooses S in the usual private key (S, F, T) to be the identity map. Thus, for UOV, the private key is only the pair (F, T) where F is the central map above, and $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an invertible linear map which is randomly chosen.

The composite map $P = F \circ T : \mathbb{F}_q^n \to \mathbb{F}_q^m$ consisting of m homogeneous quadratic polynomials in n variables over $\mathbb{F}_q$ is the public key. Note that the i-th public polynomial P_i can be written in a quadratic form, that is, $P_i = \vec{\mathbf{u}}^t [P_i] \vec{\mathbf{u}}$ where $\vec{\mathbf{u}} = (u_1, \cdots, u_n)^t$ and $[P_i] = [T]^t [F_i] [T]$ where $[T]$ is the matrix corresponding to T.

Although simple, UOV suffers extremely large public key sizes in order to be secure. Thus it is not practical if nothing new is done. It is quite a challenge to overcome it. Both NOVA and SNOVA take on the challenge by choosing the coefficients of the polynomials used in the multivariate quadratic system of UOV in a noncommutative ring, and also employs the technique of key-randomness alignment [39] and some particular designs. As a result, both NOVA and SNOVA successfully solve the problem of large public key size suffered by UOV.

One particular design in NOVA is the use of "self-canceling" perturbation technique. Although the use of perturbation trick is creative in designing, the security analysis becomes more complicated, thus we try to find an alternative way to design NOVA by skipping the perturbation trick and cook up the Simple NOVA (may also be called SNOVA where S denotes "Simple"). By skipping the perturbation trick, not only the security analysis is now more clear, but also the key generation process is shortened and both the signing and verification are accelerated. In response to the Criterion 4.B.5 of the New Call of NIST [33], we decide to submit SNOVA instead of NOVA to the New Call.

1.2 Preliminaries

1.2.1 Notations and conventions

The following Tables 1, 2 are tables that list some symbols fixed with specific meaning and some conventions on notations, respectively.

Table 1: The table of notations used in this paper.

Symbol	Description
$\mathbb{F}_q$	finite field of order q
$\mathcal{R}$	$\mathrm{Mat}_{l\times l}(\mathbb{F}_q)$, matrix ring consisting of $l \times l$ matrices over $\mathbb{F}_q$
v	number of vinegar variables
o	number of oil variables
S	symmetric matrix in $\mathcal{R}$ with irreducible characteristic polynomial
$n = v + o$	number of variables
$m = o$	number of equations
$F = [F_1, \cdots, F_m]$	central map of the signature scheme
$[F_i]$	matrix corresponding to F_i in F
T	invertible linear map in signature scheme
$[T]$	matrix corresponding to T
$[T^{-1}]$	matrix corresponding to the map T^{-1}
$P = [P_1, \cdots, P_m]$	public key of the signature scheme
$[P_i]$	matrix corresponding to P_i in P
D	document to be signed
digest	hash value of the document D, $Hash(D)$
$\mathcal{O}$	oil space of the central map F
$T^{-1}(\mathcal{O})$	oil space of the public key P
$MQ(N, M, q)$	complexity of a MQ system of M equations in N variables over $\mathbb{F}_q$

Table 2: The table of conventions in this paper.

Description	**The font denoted with**	**Example**
Integers	lower case letters	n, m and l
Elements in $\mathcal{R}$	upper case letters	A, S and Q
Variables over $\mathcal{R}$	upper case letters	$X_1, \cdots, X_n$
Elements in $\mathbb{F}_q$	lower case letters	$a_0, \cdots, a_{l-1}$
Variables over $\mathbb{F}_q$	lower case letters	$x_1, \cdots, x_n$
Vectors of any dimension	boldface letters with arrow on top	$\vec{\mathbf{X}}$ and $\vec{\mathbf{x}}$
Vector spaces and rings	calligraphic font	$\mathcal{O}$ and $\mathcal{R}$
The (j,k)-th entry of the matrix $[F_i]$, $[T]$ and $[P_i]$, respectively	subscript j,k	$F_{i,jk}$, T_{jk} and $P_{i,jk}$
Block form of matrices $[T]$	upper case letters	$[T] = \begin{bmatrix} T^{11} & T^{12} \\ T^{21} & T^{22} \end{bmatrix}$
Block form of matrices $[F_i]$	upper case letters	$[F_i] = \begin{bmatrix} F_i^{11} & F_i^{12} \\ F_i^{21} & 0 \end{bmatrix}$
Block form of matrices $[P_i]$	upper case letters	$[P_i] = \begin{bmatrix} P_i^{11} & P_i^{12} \\ P_i^{21} & P_i^{22} \end{bmatrix}$

1.2.2 Basic notions

MQ problem. Let $\mathbb{F}_q$ be a finite field of order q. Given M quadratic polynomials $P(\vec{\mathbf{x}}) = [P_1(\vec{\mathbf{x}}), \cdots, P_M(\vec{\mathbf{x}})]$ in N variables $\vec{\mathbf{x}} = (x_1, \cdots, x_N)^t$ and a vector $\vec{\mathbf{y}} \in \mathbb{F}_q^M$, to find a vector $\vec{\mathbf{u}} \in \mathbb{F}_q^N$ such that $P(\vec{\mathbf{u}}) = [P_1(\vec{\mathbf{u}}), \cdots, P_M(\vec{\mathbf{u}})] = \vec{\mathbf{y}}$. This problem is known to be NP-hard [23]. Note that, it is generically expected to be exponentially hard in the case $N \sim M$ and it can be solved in polynomial time for $M \geq \frac{N(N+1)}{2}$ or $N \geq M(M+1)$ [7].

In this paper, we use $MQ(N, M, q)$ to denote the complexity of solving such an MQ problem. There are several algorithms to solve a multivariate quadratic system of M equations in N variables over finite fields such as F_4 [20], F_5 [21] and XL variants [15, 52].

Polar forms. The polar form of a homogeneous multivariate quadratic map $P(\vec{\mathbf{x}}) = [P_1(\vec{\mathbf{x}}), \cdots, P_M(\vec{\mathbf{x}})]$, consisting of M multivariate homogeneous quadratic polynomial in n variables, is defined to be the map

$$P'(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = [P'_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}), \cdots, P'_M(\vec{\mathbf{x}}, \vec{\mathbf{y}})]$$

where the polar form of $P_i(\vec{\mathbf{x}})$ is defined by

$$P_i'(\vec{\mathbf{x}}, \vec{\mathbf{y}}) = P_i(\vec{\mathbf{x}} + \vec{\mathbf{y}}) - P_i(\vec{\mathbf{x}}) - P_i(\vec{\mathbf{y}})$$

which is symmetric and bilinear. Note that if $[P_i]$ is the matrix related to P_i, i.e., $P_i(\vec{\mathbf{x}}) = \vec{\mathbf{x}}^t [P_i] \mathbf{x}$, then the matrix related to P_i' is $[P_i'] = [P_i] + [P_i]^t$

1.2.3 NIST security level.

In [32], NIST suggested several security levels for post-quantum cryptosystem design. In the new call for additional digital signature scheme project, NIST slightly modified their security level request. Herein, we focus on levels I, III, and V. The NIST security levels I, III and V require that a classical attacker needs 2^{143}, 2^{207} and 2^{272} classical gates to break the scheme, and 2^{61}, 2^{125} and 2^{189} quantum gates for a quantum attacker, respectively.

The number of gates required for an attack against digital signature scheme can be computed by

$$\sharp\text{gates} = \sharp\text{field multiplication} \cdot (2 \cdot (\log_2 q)^2 + \log_2 q)$$

with the assumption that one field multiplication in the field $\mathbb{F}_q$ needs about $(\log_2 q)^2$ bit multiplications and same for bit additions and for each field multiplication in the computation, it also needs an addition of field elements, each takes $\log_2 q$ bit additions.

Table 3: NIST Security Level.

Security Level	Classical gates	Quantum gates
I	143	61
III	207	125
V	272	189

1.2.4 Unbalanced Oil and Vinegar signature (UOV) scheme

The Unbalanced Oil and Vinegar (UOV) signature scheme [27] signature scheme is a slight modification of the Oil and Vinegar (OV) [36] signature scheme, proposed by Patarin in 1997. This scheme is based on a trapdoor map F which is easily inverted and it also can resist the K-S attack [28] on OV.

A (v, o, q) UOV signature scheme with $v > o$ is defined with a triple of positive integers so that the number of variables $n = v + o$, the number of equations $m = o$, and over $\mathbb{F}_q$.

Central map. The central map of UOV scheme is $F = [F_1, \cdots, F_m] : \mathbb{F}_q^n \to \mathbb{F}_q^m$ where

each F_i is of the form

$$F_i = \sum_{j=1}^{v} \sum_{k=j}^{n} f_{i,jk} x_j x_k.$$

The coefficients $f_{i,jk}$'s are chosen randomly from $\mathbb{F}_q$. Note that each F_i is a homogeneous quadratic polynomials in n variables which has no terms $x_j x_k$ for $j, k = v+1, \cdots, n$ over $\mathbb{F}_q$. The variables $x_1, \cdots, x_v$ are called the vinegar variables and $x_{v+1}, \cdots, x_n$ are called the oil variables.

Private key and Public key. The private key of UOV is the pair (F, T) where $T : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an invertible linear map which is randomly chosen. The map $P = F \circ T : \mathbb{F}_q^n \to \mathbb{F}_q^m$ where $P_i = F_i \circ T$. The quadratic form of P_i is $P_i = \vec{\mathbf{u}}^t [P_i] \vec{\mathbf{u}}$ where $\vec{\mathbf{u}} = (u_1, \cdots, u_n)^t$ and $[P_i] = [T]^t [F_i] [T]$ where $[T]$ is the matrix related to T.

Oil space, $\mathcal{O}$. The special structure of F in UOV scheme indicates that F vanishes on the linear space $\mathcal{O} = \{\vec{\mathbf{x}} \in \mathbb{F}_q^n : x_1 = \cdots = x_v = 0\}$ called the oil space of central map F, and hence the oil space of public key P will be the space $T^{-1}(\mathcal{O})$.

Public key generation and drawback. A. Petzoldt [38] and Rainbow [16] of the third-round of NIST proposal realized that the part of the randomness of the private key can be transferred to the public key and then a large part of public key can be generated by a PRNG. This reduces the public key size of UOV to the order $O(m^3 \cdot \log q)$. However, the size of the public key of UOV scheme is still too large to be a practical scheme, for example, to meet the security levels I, III, and V in the PQC project of NIST [32].

1.3 Parameter space of the SNOVA scheme

The parameter set of a SNOVA scheme is completely described by a quadruple (v, o, q, l) of positive integers with induced parameters $m = o$ and $n = v + o$ as explained below.

- v is the number of vinegar variables over the noncommutative ring $\mathcal{R}$.
- o is the number of oil variables over the noncommutative ring $\mathcal{R}$ and we require that $o < v$.
- q is the order of the underlying finite field $\mathbb{F}_q$ of characteristic two.
- l indicates the size of the noncommutative ring $\mathcal{R} = \mathrm{Mat}_{l \times l}(\mathbb{F}_q)$.
- $m = o$ is the number of quadratic equations over $\mathcal{R}$ in the public key.
- $n = v + o$ is the number of variables over $\mathcal{R}$ in the public key.

1.4 Design rationale

We believe that multivariate cryptosystems are useful in cryptography. However, for multivariate cryptosystems over fields, there are abundance of cryptanalysis tools available such as F_4, F_5, XL [20, 21, 15]. Also, the problem of suffering large public key size makes their application less practical. Therefore we are determined to design multivariate cryptosystems over noncommutative rings and also to solve the problem of suffering large public key size.

Due to its simplicity, UOV [27] is an ideal test ground of these ideas. Although the idea of using noncommutative rings applies to general noncommutative rings, we decide to start from the matrix ring $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$. And also using finite fields not of characteristic two is possible, using finite fields of characteristic two enjoys the benefits of possible tricks of speeding in computation.

It would be not wise to simply generalize UOV over finite fields to over noncommutative rings, which means those skills in attacking UOV over finite fields might be applicable to UOV over noncommutative rings (may be called ring UOV). Therefore, we adopt multiplication with other random matrices before and after the ring UOV and summing them together. Also, by viewing each element in $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$ as l^2 elements in $\mathbb{F}_q$, we may regard our system as a multivariate system over finite fields but with the explosion of the number of variables and the number of equations. In doing so, the structure of multiplication of matrices may result in the sparsity of the product matrix with entries in $\mathbb{F}_q$. The multiplication with other random matrices and then summing together happens to scramble the entries in $\mathbb{F}_q$ and hence make the sparsity disappear.

The above is done with a trade-off in computation speed, which we think it is justifiable.

Also, to further solve the problem of large public key size, we find that the technique of shifting the randomness of the private key to a part of the public key [38] (which may be called key-randomness alignment) and in combination of using PRNG with seeds applicable to multivariate cryptosystems over noncommutative rings. The result is an amazing success in reducing the key sizes substantially at the same security level.

The designing process is actually a long process of "try-and-err". We have done our best to ensure the security of our design of SNOVA against all known major attacks. For details, please refer to the part of security analysis.

1.5 SNOVA signature scheme

1.5.1 Description

Let v, o be positive integers with $v > o$ and $\mathbb{F}_q$ be of characteristic 2. For example, we choose $\mathbb{F}_q = \mathrm{GF}(16)$ for our implementation. Let $n = v + o$ and $m = o$. A (v, o, q, l)

SNOVA signature scheme is defined as the following.

Subring $\mathbb{F}_q[S]$ and elements in $\mathbb{F}_q[S]$. Let S be a $l \times l$ symmetric matrix with irreducible characteristic polynomial. The subring $\mathbb{F}_q[S]$ of $\mathcal{R}$ is defined to be

$$\mathbb{F}_q[S] = \{a_0 + a_1 S + \cdots + a_{l-1} S^{l-1} \mid a_0, a_1, \cdots, a_{l-1} \in \mathbb{F}_q\}$$

and note that the elements in $\mathbb{F}_q[S]$ are also symmetric and they all commute.

Central map. Let $\Omega = \{(j,k) : 1 \leq j,k \leq n\} \setminus \{(j,k) : m+1 \leq j,k \leq n\}$. The central map of SNOVA scheme is $F = [F_1, \cdots, F_m] : \mathcal{R}^n \to \mathcal{R}^m$ and for $i = 1, \cdots, m$, F_i is defined to be

$$F_i = \sum_{\alpha=1}^{l^2} A_\alpha \cdot \left(\sum_{(j,k)\in\Omega} X_j^t \left(Q_{\alpha 1} F_{i,jk} Q_{\alpha 2}\right) X_k \right) \cdot B_\alpha$$

where $F_{i,jk}$'s are randomly chosen from $\mathcal{R}$, A_α and B_α are invertible elements randomly chosen from $\mathcal{R}$, and $Q_{\alpha 1},\ Q_{\alpha 2}$ are invertible matrices randomly chosen from $\mathbb{F}_q[S]$.

Due to the noncommutativity of matrix ring $\mathcal{R}$, the matrix $[F_i]$ over $\mathcal{R}$ corresponding to F_i is of the form

$$[F_i] = \begin{bmatrix} F_i^{11} & F_i^{12} \\ F_i^{21} & 0 \end{bmatrix},$$

where F_i^{11}, F_i^{12} and F_i^{21} are matrices over $\mathcal{R}$ of size $v \times v$, $v \times o$ and $o \times v$, respectively.

Invertible linear map. The invertible linear map in SNOVA scheme is the map $T : \mathcal{R}^n \to \mathcal{R}^n$ related to the matrix

$$[T] = \begin{bmatrix} I^{11} & T^{12} \\ 0 & I^{22} \end{bmatrix},$$

where T^{12} is a $v \times o$ matrix consisting of nonzero entries T_{ij} chosen randomly in $\mathbb{F}_q[S]$. Note that T_{ij} is symmetric and commutes with other elements in $\mathbb{F}_q[S]$. The matrices I^{11} and I^{22} are the diagonal matrices with all diagonal entries being the $l \times l$ identity matrix, i.e. the unity in $\mathcal{R}$. Therefore, $[T]$ is invertible and hence T.

Private key. The private key of SNOVA is (F, T), i.e., the matrix $[T]$ and the matrices $[F_i]$ for $i = 1, 2, \ldots, m$.

Public key. Let $P = F \circ T$ be the public key of SNOVA scheme. For $i = 1, 2, \ldots, m$, $P_i = F_i \circ T$. The relation $\vec{\mathbf{X}} = [T] \cdot \vec{\mathbf{U}}$ with the variable $\vec{\mathbf{U}} = (U_1, \cdots, U_n)^t$ implies that

$$P_i(\vec{\mathbf{U}}) = F_i(T(\vec{\mathbf{U}})) = \sum_{\alpha=1}^{l^2} \sum_{d_j=1}^{n} \sum_{d_k=1}^{n} A_\alpha \cdot U_{d_j}^t (Q_{\alpha 1} P_{i,d_j d_k} Q_{\alpha 2}) U_{d_k} \cdot B_\alpha$$

where $P_{i,d_j d_k} = \sum_{\Omega} T_{j,d_j} \cdot F_{i,jk} \cdot T_{k,d_k}$ by the commutativity of $\mathbb{F}_q[S]$. Therefore, the public key consists of the corresponding matrices

$$[P_i] = [T]^t [F_i] [T],\ i = 1, \cdots, m$$

and the matrices $A_\alpha,\ B_\alpha$ and $Q_{\alpha k}$ for $\alpha = 1, 2, \ldots, l^2$ and $k = 1, 2$.

Signature. Let D be the document to be signed and $Hash(D) = \vec{\mathbf{Y}} = (Y_1, \cdots, Y_m)^t \in \mathcal{R}^m$ be its hash value. We compute the signature $\vec{\mathbf{U}}$ step by step. First, we assign values to vinegar variables $X_1, \cdots, X_v$ randomly and the resulting system can be seen as a linear system over the $\mathbb{F}_q$-entries of oil variables $X_{v+1}, \cdots, X_n$. The remaining is the same as in UOV scheme. Secondly, the signature is $\vec{\mathbf{U}} = T^{-1}(\vec{\mathbf{X}}) \in \mathcal{R}^n$.

Verification. Let $\vec{\mathbf{U}} = (U_1, \cdots, U_n)^t \in \mathcal{R}^n$ be the signature to be verified. If $Hash(D) = P(\vec{\mathbf{U}})$, then the signature is accepted, otherwise rejected.

Structure of SNOVA. A (v, o, q, l) SNOVA over $\mathcal{R}$ can be regarded as an (l^2v, l^2o, q) UOV scheme over $\mathbb{F}_q$. The noncommutativity of matrix ring $\mathcal{R}$ implies that we can not write the public key of SNOVA into a quadratic form $P_i(\vec{\mathbf{U}}) = (\vec{\mathbf{U}})^t [P_i] \vec{\mathbf{U}}$ over $\mathcal{R}$.

1.5.2 Key generation process of SNOVA

We give the standard key generation process of SNOVA and the key generation process with key-randomness alignment technique. Note that in SNOVA scheme, $\mathbb{F}_q$ is of the characteristic 2.

Standard key generation process. For $i = 1, 2, \ldots, m$, the matrix $[P_i]$ is obtained by relation

$$[T]^t [F_i] [T] = [P_i] = \begin{bmatrix} P_i^{11} & P_i^{12} \\ P_i^{21} & P_i^{22} \end{bmatrix}.$$

Then, we have the following

$$\begin{aligned} P_i^{11} &= F_i^{11} \\ P_i^{12} &= F_i^{11} T^{12} + F_i^{12} \\ P_i^{21} &= (T^{12})^t F_i^{11} + F_i^{21} \\ P_i^{22} &= (T^{12})^t \cdot \left(F_i^{11} T^{12} + F_i^{12}\right) + F_i^{21} T^{12}. \end{aligned}$$

Therefore, to generate the public key we generate the matrices $[F_i]$, $[T]$ from a seed $\mathbf{s_{private}}$ at first and then compute the public key $[P_i]$ for $i = 1, \cdots, m$ with the formulas above.

Key generation with randomness alignment. The following are steps of key generation process of SNOVA with key randomness alignment.

First Step: Fix an $l \times l$ symmetric matrix S with irreducible characteristic polynomial. Generate $P_i^{11},\ P_i^{12}$ and P_i^{21} for $i = 1, \cdots, m$ from public seed $\mathbf{s_{public}}$. Generate $[T]$ from private seed $\mathbf{s_{private}}$. We also generate the matrices $A_\alpha,\ B_\alpha,\ Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $\alpha = 1, 2, \ldots, l^2$ from $\mathbf{s_{public}}$.

Second Step: Compute the matrix $F_i^{11}, F_i^{12}, F_i^{21}, P_i^{22}$ for $i = 1, \cdots, m$ as below.

For $i = 1, 2, \ldots, m$, we have

$$[F_i] = \left[T^{-1}\right]^t [P_i] \left[T^{-1}\right].$$

Therefore, the following equations hold

$$\begin{aligned} F_i^{11} &= P_i^{11} \\ F_i^{12} &= P_i^{11} T^{12} + P_i^{12} \\ F_i^{21} &= (T^{12})^t P_i^{11} + P_i^{21} \\ 0 = F_i^{22} &= (T^{12})^t \cdot \left(P_i^{11} T^{12} + P_i^{12}\right) + P_i^{21} T^{12} + P_i^{22}. \end{aligned}$$

In other words, we then have

$$P_i^{22} = \left(T^{12}\right)^t \cdot \left(P_i^{11} T^{12} + P_i^{12}\right) + P_i^{21} T^{12}.$$

1.5.3 To attain EUF-CMA security

For practical considerations, we use a random binary vector, called salt in order to achieve Existential Unforgeability under Chosen Message Attack (EUF-CMA) Security [34, 41].

Signature. Let D be the document to be sign, we randomly choose **salt** and then generate a signature for the hash value $\vec{\mathbf{Y}} = Hash(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$ where $\mathbf{s_{public}}$ is the seed used to generate the public key of SNOVA scheme.

Therefore, the corresponding signature with salt is of the form $\vec{\sigma} = (\vec{\mathbf{U}}||\mathbf{salt})$ where $\vec{\mathbf{U}}$ is the signature of $\vec{\mathbf{Y}}$ generated by the SNOVA signer. Note that we want almost no **salt** is used for more than one signature. Therefore, the length of **salt** is chosen to be 16 Bytes under the assumption of up to 2^{64} signatures being generated with the system [33].

Verification. If $P(\vec{\mathbf{U}}) = Hash(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$, the signature is accepted, otherwise rejected.

1.6 Implementation details

In this section, we describe the details about implementations.

For all parameter settings, we implement two variants of SNOVA scheme:

- SNOVA-ssk: In this variant, the suffix "ssk" stand for "seed-type secret key". Secret key only stores the information of seeds. This means the private key expansion and the expansion of the random part of the public key are included

in both the key generation procedure and the signing procedure of the signer. In other words, the **Algorithm**s 3, 4, 6 are part of both key generation process and signing process.

- SNOVA-esk: In this variant, the suffix "esk" represents "expanded secret key". Therefore, the private key expansion is only a part of key generation procedure and then the expanded secret key is stored and directly accessed when the signer intend to sign. That is, the **Algorithm**s 3, 4, 6 are only used in key generation process.

In SNOVA scheme, several hash functions and PRNG are needed. We categorize the parts that needed hash functions and PRNG and explain which instance we take in each case:

- The length of private key seed, $|\mathbf{s_{private}}|$: 32 bytes.
- The length of public key seed, $|\mathbf{s_{public}}|$: 16 bytes.
- The length of **salt**, $|\mathbf{salt}|$: 16 bytes.
- The hash function which is used to generate private key T: SHAKE256.
- The hash function which is used to generate the random part of public key: AES128.
- The digest of the document D: $\mathbf{digest} = Hash(D)$.
- The hash function which is used to to generate the hash value to be signed $Hash_{\text{SHAKE256}}(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$: SHAKE256.
- The hash function which is used to generate vinegar values we used in signature generation, $Hash_{\text{SHAKE256}}(\mathbf{s_{private}}||\mathbf{digest}||\mathbf{salt}||\mathbf{num_{sig}})$: SHAKE256.

To generate the longer random part of the public key efficiently, we have adopted AES128-CTR encryption. This involves using the public key seed as the encryption key and encrypting a zero plaintext block with a zero nonce. The resulting ciphertext serves as the pseudo-random output for generating the random part of the public key.

It is important to note that the provided implementation uses OpenSSL's AES128-CTR. When implementing this method, it is crucial to follow cryptographic security best practices. By utilizing trusted cryptographic libraries, we can ensure proper implementation of the encryption algorithm and enhance security.

In the future, we may explore alternative methods for generating the random part of the public key if more efficient implementation approaches are discovered.

1.7 Constants and tables

The finite field $\mathbb{F}_{16}$. Fix an irreducible polynomial $f(x) = x^4 + x + 1$ over $\mathbb{F}_2$ and consider that the finite field $\mathbb{F}_{16}$ consists of the polynomials $ax^3 + bx^2 + cx + d \in \mathbb{F}_2[x]$ modulo $f(x)$. The elements of $\mathbb{F}_{16}$ are stored in 4 bits and then the addition of two elements in $\mathbb{F}_{16}$ is equal to the bitwise XOR of them. For simplicity, we convert binary elements of $\mathbb{F}_{16}$ to decimal numbers. In particular, an element $ax^3 + bx^2 + cx + d$ of $\mathbb{F}_{16}$ is converted to an integer $2^3a + 2^2b + 2c + d$. For multiplications of $\mathbb{F}_{16}$, we fix a generator 2 of the multiplicative group $\mathbb{F}_{16}^{\times}$ and create a list

$$\mathbf{F}^{\times} := \{2^i \mid 0 \leq i \leq 15\} = \{1, 2, 4, 8, 3, 6, 12, 11, 5, 10, 7, 14, 15, 13, 9\}.$$

Then we create a multiplication table **mt** of $\mathbb{F}_{16}$ as follows

$$\begin{aligned}
&\mathbf{mt}(\mathbf{F}^{\times}[i], 0) = \mathbf{mt}(0, \mathbf{F}^{\times}[i]) := 0 \quad \text{for } 1 \leq i \leq 15 \text{ and} \\
&\mathbf{mt}(\mathbf{F}^{\times}[i], \mathbf{F}^{\times}[j]) := \mathbf{F}^{\times}[i + j \pmod{15}] \quad \text{for } 1 \leq i, j \leq 15.
\end{aligned}$$

The matrix S. When we fix the finite field $\mathbb{F}_{16} := \mathbb{F}_2[x]/ < x^4 + x + 1 >$ and with the same notation as above, we fix the matrices S as follows:

$$S = \begin{bmatrix} 8 & 7 \\ 7 & 6 \end{bmatrix} \quad \text{if } l = 2,$$

$$S = \begin{bmatrix} 8 & 7 & 6 \\ 7 & 6 & 5 \\ 6 & 5 & 4 \end{bmatrix} \quad \text{if } l = 3,$$

$$S = \begin{bmatrix} 8 & 7 & 6 & 5 \\ 7 & 6 & 5 & 4 \\ 6 & 5 & 4 & 3 \\ 5 & 4 & 3 & 2 \end{bmatrix} \quad \text{if } l = 4.$$

One can check that the characteristic polynomials of these matrices S are irreducible over $\mathbb{F}_2$.

Generate elements of $\mathbb{F}_{16}[S]$. Recall that the subring $\mathbb{F}_{16}[S]$ of $\mathcal{R}$ is defined to be

$$\mathbb{F}_{16}[S] = \{a_0 + a_1 S + \cdots + a_{l-1} S^{l-1} \mid a_0, a_1, \cdots, a_{l-1} \in \mathbb{F}_{16}\}$$

and the entries of the matrix T^{12} are nonzero matrices randomly chosen from $\mathbb{F}_{16}[S]$. In order to generate nonzero matrices from $\mathbb{F}_{16}[S]$, we will modify the leading coefficient a_{l-1} if $a_{l-1} = 0$. Given inputs l elements $a_0, \ldots, a_{l-1}$ of $\mathbb{F}_{16}$. If $a_{l-1} = 0$, then we modify the leading coefficient $a_{l-1} := 16 - a_0$ when $a_0 \neq 0$ and $a_{l-1} := 15$ when $a_0 = 0$. Note that $16 - a_0$ is a difference between two integers and it is not compatible with the difference between elements of $\mathbb{F}_{16}$.

Algorithm 1: Generate elements of $\mathbb{F}_{16}[S]$

input : l elements $a_0, \ldots, a_{l-1}$ of $\mathbb{F}_{16}$
output: a nonzero element of $\mathbb{F}_{16}[S]$

```
if a_{l-1} = 0 then
    if a_0 ≠ 0 then
        a_{l-1} ← 16 − a_0
    else
        if a_0 = 0 then
            a_{l-1} ← 15
        end
    end
end
return a_0 + a_1 S + ··· + a_{l-1} S^{l-1}
```

Generate invertible matrices. Let $l = 2, 3$ or 4 and $M \in \mathrm{Mat}_{l\times l}(\mathbb{F}_{16})$ any $l \times l$ matrix over $\mathbb{F}_{16}$. Since the polynomial $\det(M + xS)$ in the variable x has at most l roots, there exists an element a of $\mathbb{F}_{16}$ such that the matrix $M + aS$ is invertible. We use this property to generate invertible matrices as follows.

Algorithm 2: Generate invertible matrices

input : a $l \times l$ matrix M
output: an invertible matrix M

```
if det(M) = 0 then
    for a from 1 to 15 do
        if det(M + aS) ≠ 0 then
            M ← M + aS
            break
        end
    end
end
return M
```

1.8 Algorithms

1.8.1 Algorithms for key generation

For convenience, we always start the index with zero in our algorithms. The key generation process with key-randomness alignment technique is as follows.

First Step: Fix an $l \times l$ symmetric matrix S as in Sec. 1.7. Generate $[T]$ from private seed $\mathbf{s_{private}}$. Generate P_i^{11}, P_i^{12} and P_i^{21} for $0 \le i < m$ from public seed $\mathbf{s_{public}}$. We also generate the matrices A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $0 \le \alpha < l^2$, and P_i^{11}, P_i^{12} and P_i^{21} for $0 \le i < m$ from $\mathbf{s_{public}}$.

Algorithm 3: Generate the linear map T

input : SNOVA parameters (v, o, l)
private seed $\mathbf{s_{private}}$
output: the matrix $[T^{12}]$

(coefficients of S-polynomials for entries in T^{12})$\leftarrow Hash_{\text{SHAKE256}}(\mathbf{s_{private}})$
▷ $Hash_{\text{SHAKE256}}$ is instantiated as SHAKE256 throughout

Generate entries of T^{12} using **Algorithm** 1

return $[T^{12}]$

Algorithm 4: Generate the random part of public key

input : SNOVA parameters (v, o, l)
public seed $\mathbf{s_{public}}$
output: the matrices (A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $0 \leq \alpha < l^2$)
the matrices (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)
(entries of $(P_0^{11}||\cdots||P_{m-1}^{11})||(P_0^{12}||\cdots||P_{m-1}^{12})||(P_0^{21}||\cdots||P_{m-1}^{21}))||$
(entries of $(A_0||\cdots||A_{l^2-1})||(B_0||\cdots||B_{l^2-1}))||$

(coefficients of S-polynomials for entries in $(Q_{01}||\cdots||Q_{(l^2-1)1})||(Q_{02}||\cdots||Q_{(l^2-1)2}))$
$\leftarrow Hash_{\text{AES128}}(\mathbf{s_{public}})$
▷ $Hash_{\text{AES128}}$ is instantiated as AES128 throughout

for α *from* 0 *to* $l^2 - 1$ **do**
  let A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ be invertible using **Algorithm** 2
end
return (A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $0 \leq \alpha < l^2$) and (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)

Second Step: Compute the matrix $F_i^{11}, F_i^{12}, F_i^{21}, P_i^{22}$ for $0 \leq i < m$ as below.

For $0 \leq i < m$, we have

$$[F_i] = \left[T^{-1}\right]^t [P_i] \left[T^{-1}\right].$$

Note that $[T^{-1}] = [T]^{-1} = [T]$ since $\mathbb{F}_{16}$ is of the characteristic 2. Therefore, the following equations hold

$$\begin{aligned} F_i^{11} &= P_i^{11} \\ F_i^{12} &= P_i^{11} T^{12} + P_i^{12} \\ F_i^{21} &= (T^{12})^t P_i^{11} + P_i^{21} \\ 0 = F_i^{22} &= (T^{12})^t \cdot \left(P_i^{11} T^{12} + P_i^{12}\right) + P_i^{21} T^{12} + P_i^{22}. \end{aligned}$$

In other words, we then have

$$P_i^{22} = \left(T^{12}\right)^t \cdot \left(P_i^{11} T^{12} + P_i^{12}\right) + P_i^{21} T^{12}.$$

Algorithm 5: Generate Public key

input : SNOVA parameters (v, o, l)
public and private seeds $(\mathbf{s_{public}}, \mathbf{s_{private}})$
output: public key $(\mathbf{s_{public}}, P_i^{22})$

Generate T^{12} using **Algorithm** 3
$m \leftarrow o$
Generate $(P_i^{11}, P_i^{12}, P_i^{21}$ for $0 \leq i < m)$ using **Algorithm** 4
for i from 0 to $m-1$ **do**
  $P_i^{22} \leftarrow (T^{12})^t \cdot (P_i^{11} T^{12} + P_i^{12}) + P_i^{21} T^{12}$
end
return $(\mathbf{s_{public}}, P_i^{22}$ for $0 \leq i < m)$

Algorithm 6: Generate private key

input : SNOVA parameters (v, o, l)
public and private seeds $(\mathbf{s_{public}}, \mathbf{s_{private}})$
output: private key $(T^{12}, F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \leq i < m)$

Generate T^{12} using **Algorithm** 3
$m \leftarrow o$
Generate $(P_i^{11}, P_i^{12}, P_i^{21}$ for $0 \leq i < m)$ using **Algorithm** 4
for i *from* 0 *to* $m-1$ **do**
  $F_i^{11} \leftarrow P_i^{11}$
  $F_i^{12} \leftarrow P_i^{11} T^{12} + P_i^{12}$
  $F_i^{21} \leftarrow (T^{12})^t P_i^{11} + P_i^{21}$
end
return $(T^{12}, F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \leq i < m)$

1.8.2 Algorithms for signature generation

Let D be the document to be signed, we randomly choose **salt** and then generate a signature for the hash value $\vec{\mathbf{Y}} = Hash(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$ where $\mathbf{s_{public}}$ is the public seed of SNOVA scheme. First, we randomly assign values to vinegar variables $X_0, \cdots, X_{v-1}$ depending on the number of sign $\mathbf{num_{sig}}$ as follows

Algorithm 7: Assign values to vinegar variables

input : SNOVA parameters (v, o, l)
digest of the document $\mathbf{digest} = Hash(D)$
the number of sign $\mathbf{num_{sig}}$
salt
output: vinegar values $(X_0, \ldots, X_{v-1})$

$(X_0, \ldots, X_{v-1}) \leftarrow Hash(\mathbf{s_{private}}||\mathbf{digest}||\mathbf{salt}||\mathbf{num_{sig}})$
return $(X_0, \ldots, X_{v-1})$

Second, we compute the vinegar part values $F_{i,VV}$ of the central map F_i for $0 \leq i < m$. Recall that the vinegar part values $F_{i,VV}$ of the central map F_i is

$$F_{i,VV} = \sum_{\alpha=0}^{l^2-1} A_\alpha \cdot \left(\sum_{j=0}^{v-1} \sum_{k=0}^{v-1} X_j^t \left(Q_{\alpha 1} F_{i,jk} Q_{\alpha 2} \right) X_k \right) \cdot B_\alpha$$

where $F_{i,jk}$'s are elements randomly chosen from $\mathcal{R}$, A_α and B_α are invertible matrices randomly chosen from $\mathcal{R}$, and $Q_{\alpha 1},\ Q_{\alpha 2}$ are invertible matrices randomly chosen from $\mathbb{F}_q[S]$. Note that we write the central map F_i as of the form

$$[F_i] = \begin{bmatrix} F_i^{11} & F_i^{12} \\ F_i^{21} & 0 \end{bmatrix},$$

and we also write

$$F_i^{11}[j][k] = F_{i,jk} \quad \text{for } 0 \leq j, k < v.$$

We now can compute the vinegar part values $F_{i,VV}$ of the central map F_i by the following algorithm.

Algorithm 8: Compute the vinegar part of the central map

input : SNOVA parameters (v, o, l)
private key (F_i^{11} for $0 \leq i < m$)
public key ($A_\alpha, B_\alpha, Q_{\alpha 1}, Q_{\alpha 2}$ for $0 \leq \alpha < l^2$)
vinegar values $(X_0, \ldots, X_{v-1})$
output: the vinegar part ($F_{i,VV}$ for $0 \leq i < m$)

for α from 0 to $l^2 - 1$ **do**
  for j *for* 0 *to* $v - 1$ **do**
    $\mathbf{Left}_\alpha[j] \leftarrow A_\alpha * X_j^t * Q_{\alpha 1}$ ▷ the left term of $F_i^{11}[j][k]$
    $\mathbf{Right}_\alpha[j] \leftarrow Q_{\alpha 2} * X_j * B_\alpha$ ▷ the right term of $F_i^{11}[j][k]$
  end
end
$m \leftarrow o$
for i from 0 to $m - 1$ **do**
  $F_{i,VV} \leftarrow 0$
  for α from 0 to $l^2 - 1$ **do**
    for j from 0 to $v - 1$ **do**
      for k from 0 to $v - 1$ **do**
        $F_{i,VV} \leftarrow F_{i,VV} + \mathbf{Left}_\alpha[j] * F_i^{11}[j][k] * \mathbf{Right}_\alpha[k]$
      end
    end
  end
end
return ($F_{i,VV}$ for $0 \leq i < m$)

The resulting system can be seen as a linear system of the oil variables over the finite filed $\mathbb{F}_q$. In order to write down this linear system, we need to define the vectorization

of a matrix. For $M = (m_{ij})_{l\times l} \in \mathcal{R}$ with $m_{ij} \in \mathbb{F}_q$, the vectorization of the matrix M is defined by

$$\overrightarrow{M} = \left(m_{00}, m_{01}, \cdots, m_{0(l-1)}, m_{11}, m_{12}, \cdots, m_{1(l-1)}, \cdots, m_{(l-1)(l-1)}\right)^t \in \mathbb{F}_q^{l^2}.$$

For convenience, we also write $M[i][j]$ instead of m_{ij}. Let $\mathbf{L} = (\mathbf{l}_{t_it_j})_{0\le t_i,t_j<l}$ and $\mathbf{R} = (\mathbf{r}_{t_it_j})_{0\le t_i,t_j<l} \in \mathcal{R}$. Let $X = (x_{t_it_j})_{0\le t_i,t_j<l}$ be a $l \times l$ matrix with variables $x_{t_it_j}$. Write

$$\overrightarrow{\mathbf{L}X\mathbf{R}} = M\overrightarrow{X},$$

where $M = \left(m_{t_it_j}\right)_{0\le t_i,t_j<l^2} \in \mathrm{Mat}_{l^2\times l^2}(\mathbb{F}_q)$. We find nice formulas between matrices $\mathbf{L}, \mathbf{R}$ and the matrix M. For example, $l = 2$, one has

$$\overrightarrow{\mathbf{L}X\mathbf{R}} = \begin{bmatrix} \mathbf{r}_{00}\mathbf{l}_{00}x_{00} + \mathbf{r}_{10}\mathbf{l}_{00}x_{01} + \mathbf{r}_{00}\mathbf{l}_{01}x_{10} + \mathbf{r}_{10}\mathbf{l}_{01}x_{11} \\ \mathbf{r}_{01}\mathbf{l}_{00}x_{00} + \mathbf{r}_{11}\mathbf{l}_{00}x_{01} + \mathbf{r}_{01}\mathbf{l}_{01}x_{10} + \mathbf{r}_{11}\mathbf{l}_{01}x_{11} \\ \mathbf{r}_{00}\mathbf{l}_{10}x_{00} + \mathbf{r}_{10}\mathbf{l}_{10}x_{01} + \mathbf{r}_{00}\mathbf{l}_{11}x_{10} + \mathbf{r}_{10}\mathbf{l}_{11}x_{11} \\ \mathbf{r}_{01}\mathbf{l}_{10}x_{00} + \mathbf{r}_{11}\mathbf{l}_{10}x_{01} + \mathbf{r}_{01}\mathbf{l}_{11}x_{10} + \mathbf{r}_{11}\mathbf{l}_{11}x_{11} \end{bmatrix},$$

and then

$$M = \begin{bmatrix} \mathbf{r}_{00}\mathbf{l}_{00} & \mathbf{r}_{10}\mathbf{l}_{00} & \mathbf{r}_{00}\mathbf{l}_{01} & \mathbf{r}_{10}\mathbf{l}_{01} \\ \mathbf{r}_{01}\mathbf{l}_{00} & \mathbf{r}_{11}\mathbf{l}_{00} & \mathbf{r}_{01}\mathbf{l}_{01} & \mathbf{r}_{11}\mathbf{l}_{01} \\ \mathbf{r}_{00}\mathbf{l}_{10} & \mathbf{r}_{10}\mathbf{l}_{10} & \mathbf{r}_{00}\mathbf{l}_{11} & \mathbf{r}_{10}\mathbf{l}_{11} \\ \mathbf{r}_{01}\mathbf{l}_{10} & \mathbf{r}_{11}\mathbf{l}_{10} & \mathbf{r}_{01}\mathbf{l}_{11} & \mathbf{r}_{11}\mathbf{l}_{11} \end{bmatrix}.$$

For $l = 2$, one has

$$M[t_i][t_j] = \mathbf{L}[t_i/l][t_j/l]\mathbf{R}[t_j\%l][t_i\%l] \quad \text{for } 0 \le t_i, t_j < l^2, \tag{1.1}$$

where t/l and $t\%l$ denote the quotient and the remainder of the division of t by l. In fact, the equation (1.1) holds for all l. Similarly, if we write

$$\overrightarrow{\mathbf{L}X^t\mathbf{R}} = M\overrightarrow{X},$$

then one can compute directly that

$$M[t_i][t_j] = \mathbf{L}[t_i/l][t_j\%l]\mathbf{R}[t_j/l][t_i\%l] \quad \text{for } 0 \le t_i, t_j < l^2. \tag{1.2}$$

Recall that the central map F_i including the oil variable X_k is of the form

$$\sum_{\alpha=0}^{l^2-1} A_\alpha \cdot \left(\sum_{j=0}^{v-1} X_j^t \left(Q_{\alpha 1} F_{i,jk} Q_{\alpha 2}\right) X_k\right) \cdot B_\alpha + \sum_{\alpha=0}^{l^2-1} A_\alpha \cdot \left(\sum_{j=0}^{v-1} X_k^t \left(Q_{\alpha 1} F_{i,kj} Q_{\alpha 2}\right) X_j\right) \cdot B_\alpha \tag{1.3}$$

We now use the equations (1.1), (1.2) to find the coefficient matrix M_{ik} of the variables $\overrightarrow{X_k}$ in the central map F_i as follows.

Algorithm 9: Compute the coefficient matrix of the oil variable

input : SNOVA parameters (v, o, l)
private key (F_i^{12}, F_i^{21}) for some $0 \leq i < v$
public key $(A_\alpha, B_\alpha, Q_{\alpha 1}, Q_{\alpha 2}$ for $0 \leq \alpha < l^2)$
vinegar values $(X_0, \ldots, X_{v-1})$
k, the index of oil variables where $0 \leq k < o$

output: the coefficient matrix M_{ik}

for α from 0 to $l^2 - 1$ **do**
  for j *for* 0 *to* $v - 1$ **do**
    $\mathbf{Left}_\alpha[j] \leftarrow A_\alpha * X_j^t * Q_{\alpha 1}$ ▷ the left term of $F_{i,jk}$
    $\mathbf{Right}_\alpha[j] \leftarrow Q_{\alpha 2} * X_j * B_\alpha$ ▷ the right term of $F_{i,jk}$
  end
end
for t_i from 0 to $l^2 - 1$ **do**
  for t_j from 0 to $l^2 - 1$ **do**
    $M_{ik}[t_i][t_j] \leftarrow 0$
  end
end
for α from 0 to $l^2 - 1$ **do**
  for j *for* 0 *to* $v - 1$ **do**
    $\mathbf{Left}_{X_k} \leftarrow \mathbf{Left}_\alpha[j] * F_i^{12}[j][k] * Q_{\alpha 2}$ ▷ the left term of X_k
    $\mathbf{Right}_{X_k} \leftarrow B_\alpha$ ▷ the right term of X_k
    for t_i from 0 to $l^2 - 1$ **do**
      for t_j from 0 to $l^2 - 1$ **do**
        $M_{ik}[t_i][t_j] \leftarrow M_{ik}[t_i][t_j] + \mathbf{Left}_{X_k}[t_i/l][t_j/l] * \mathbf{Right}_{X_k}[t_j\%l][t_i\%l]$
      end
    end
  end
end
for α from 0 to $l^2 - 1$ **do**
  for j *for* 0 *to* $v - 1$ **do**
    $\mathbf{Left}_{X_k} \leftarrow A_\alpha$ ▷ the left term of X_k^t
    $\mathbf{Right}_{X_k} \leftarrow Q_{\alpha 1} * F_i^{21}[k][j] * \mathbf{Right}_\alpha[j]$ ▷ the right term of X_k^t
    for t_i from 0 to $l^2 - 1$ **do**
      for t_j from 0 to $l^2 - 1$ **do**
        $M_{ik}[t_i][t_j] \leftarrow M_{ik}[t_i][t_j] + \mathbf{Left}_{X_k}[t_i/l][t_j\%l] * \mathbf{Right}_{X_k}[t_j/l][t_i\%l]$
      end
    end
  end
end
return M_{ik}

We are now ready to write down the linear system of the oil variables over the finite filed $\mathbb{F}_q$. We put the coefficient matrices M_{ik} and the vinegar part values $F_{i,VV}$ of the

central map into the augmented matrix G of the system as follows.

Algorithm 10: Build the augmented matrix of the system

input : SNOVA parameters (v, o, l)
the vinegar part values ($F_{i,VV}$ for $0 \leq i < v$)
the coefficient matrices (M_{ik} for $0 \leq i < v$ and $0 \leq k < o$)
digest of the document $\mathbf{digest} = Hash(D)$
length of the digest $|\mathbf{digest}|$
public seed $\mathbf{s_{public}}$
salt
output: the augmented matrix G

$m \leftarrow o$
$(G[0][m * l^2], \ldots, G[m * l^2 - 1][m * l^2]) \leftarrow Hash_{SHAKE256}(\mathbf{s_{public}}||\mathbf{digest}||\mathbf{salt})$
▷ Put the hash value in the last column of G
for i from 0 to $m - 1$ **do**
  for j from 0 to $l - 1$ **do**
    for k from 0 to $l - 1$ **do**
      $G[i * l^2 + j * l + k][m * l^2] \leftarrow G[i * l^2 + j * l + k][m * l^2] + F_{i,VV}[j][k]$
    end
  end
end
for i from 0 to $m - 1$ **do**
  for k from 0 to $m - 1$ **do**
    for t_i from 0 to $l^2 - 1$ **do**
      for t_j from 0 to $l^2 - 1$ **do**
        $G[i * l^2 + t_i][k * l^2 + t_j] \leftarrow M_{ik}[t_i][t_j]$
      end
    end
  end
end
return G

In the signature algorithm, we will use Gaussian elimination to solve the linear system G. For convenience, we define the function **Gauss** as follows. The function **Gauss**(G) returns a binary value $\mathbf{flag\_redo} \in \{\mathbf{TRUE}, \mathbf{FALSE}\}$ indicating whether the sign procedure needs to sign again by assigning a different set of vinegar values, and if not so **Gauss**(G) also returns the solution of the system.

Algorithm 11: Signing

input : SNOVA parameters (v, o, l)
public and private seeds $(\mathbf{s_{public}}, \mathbf{s_{private}})$
digest of the document $\mathbf{digest} = Hash(D)$
length of the digest $|\mathbf{digest}|$
salt

output: the signature **sig** and **salt**

Generate A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $0 \leq \alpha < l^2$) using Algorithm 4
$m \leftarrow o$
Generate (T^{12}, $F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \leq i < m$) using Algorithm 6
$[T] \leftarrow \begin{bmatrix} I^{11} & T^{12} \\ 0 & I^{22} \end{bmatrix}$
$\mathbf{num_{sig}} \leftarrow 0$
repeat
  $\mathbf{num_{sig}} \leftarrow \mathbf{num_{sig}} + 1$
  Assign vinegar values $(X_0, \ldots, X_{v-1})$ using Algorithm 7
  Compute ($F_{i,VV}$ for $0 \leq i < m$) using Algorithm 8
  Compute (M_{ik} for $0 \leq i < m$, $0 \leq k < o$) using Algorithm 9
  Build the augmented matrix G using Algorithm 10
  flag_redo, $(\widetilde{X}_0, \widetilde{X}_1, \ldots, \widetilde{X}_{o-1}) \leftarrow$ **Gauss**(G)
  if flag_redo==FALSE then
    sig $\leftarrow [T](X_0, \ldots, X_{v-1}, \widetilde{X}_0, \ldots, \widetilde{X}_{o-1})^t$ ▷ Note that $T^{-1} = T$
  end
until flag_redo==FALSE;
return (**sig**, **salt**)

1.8.3 Algorithms for signature verification

Recall that the public key $P = [P_0, \cdots, P_{m-1}] : \mathcal{R}^n \to \mathcal{R}^m$, where

$$P_i(\vec{\mathbf{U}}) = \sum_{\alpha=0}^{l^2-1} \sum_{d_j=0}^{n-1} \sum_{d_k=0}^{n-1} A_\alpha \cdot U_{d_j}^t (Q_{\alpha 1} P_{i,d_j d_k} Q_{\alpha 2}) U_{d_k} \cdot B_\alpha$$

with the variable $\vec{\mathbf{U}} = (U_0, \ldots, U_{m-1})^t$. For the signature verification, we write the signature $\mathbf{sig} = (U_0, \ldots, U_{m-1})^t \in \mathcal{R}$ and $\mathbf{sig}[i] = U_i$ for $0 \leq i < m$. The algorithm for evaluating the public map at a signature **sig** is as follows.

Algorithm 12: Evaluate the public map

input : SNOVA parameters (v, o, l)
public key $(A_\alpha, B_\alpha, Q_{\alpha 1}, Q_{\alpha 2}$ for $0 \leq \alpha < l^2)$
public map $(P_i^{11}, P_i^{12}, P_i^{21}, P_i^{22}$ for $0 \leq i < m)$
the signature **sig**

output: The evaluation $\mathbf{hash_s}$ of P at **sig**

$m \leftarrow o$
for α from 0 to $m-1$ **do**
  for j from 0 to $n-1$ **do**
    $\mathbf{Left}_\alpha[j] \leftarrow A_\alpha * (\mathbf{sig}[j])^t * Q_{\alpha 1}$ ▷ the left term of $P_{i,d_j d_k}$
    $\mathbf{Right}_\alpha[j] \leftarrow Q_{\alpha 2} * \mathbf{sig}[j] * B_\alpha$ ▷ the right term of $P_{i,d_j d_k}$
  end
end
for i from 0 to $m-1$ **do**
  $\mathbf{hash_s}[i] \leftarrow 0$
  for α from 0 to l^2-1 **do**
    for d_j from 0 to $v-1$ **do**
      for d_k from 0 to $v-1$ **do**
        $\mathbf{hash_s}[i] = \mathbf{hash_s}[i] + \mathbf{Left}_\alpha[d_j] * P_i^{11}[d_j][d_k] * \mathbf{Right}_\alpha[d_k]$
      end
    end
    for d_j from 0 to $v-1$ **do**
      for d_k from 0 to $o-1$ **do**
        $\mathbf{hash_s}[i] = \mathbf{hash_s}[i] + \mathbf{Left}_\alpha[d_j] * P_i^{12}[d_j][d_k] * \mathbf{Right}_\alpha[v+d_k]$
      end
    end
    for d_j from 0 to $o-1$ **do**
      for d_k from 0 to $v-1$ **do**
        $\mathbf{hash_s}[i] = \mathbf{hash_s}[i] + \mathbf{Left}_\alpha[v+d_j] * P_i^{21}[d_j][d_k] * \mathbf{Right}_\alpha[d_k]$
      end
    end
    for d_j from 0 to $o-1$ **do**
      for d_k from 0 to $o-1$ **do**
        $\mathbf{hash_s}[i] = \mathbf{hash_s}[i] + \mathbf{Left}_\alpha[v+d_j] * P_i^{22}[d_j][d_k] * \mathbf{Right}_\alpha[v+d_k]$
      end
    end
  end
end
$\mathbf{hash_s} \leftarrow (\mathbf{hash_s}[0], \ldots, \mathbf{hash_s}[m-1])^t$
return $\mathbf{hash_s}$

Algorithm 13: Signature verification

input : SNOVA parameters (v, o, l)
public key $(\mathbf{s_{public}}, P_i^{22}$ for $0 \leq i < m)$
digest of the document $\mathbf{digest} = Hash(D)$
length of the digest $|\mathbf{digest}|$
salt

output: **Accept** or **Reject**

Generate A_α, B_α, $Q_{\alpha 1}$ and $Q_{\alpha 2}$ for $0 \leq \alpha < l^2$) using Algorithm 4
$m \leftarrow o$
Generate $(P_i^{11}, P_i^{12}, P_i^{21}$ for $0 \leq i < m)$ using Algorithm 4
$\mathbf{hash_d} \leftarrow Hash_{SHAKE256}(\mathbf{s_{public}}||\mathbf{digest}||\mathbf{salt})$
Compute $\mathbf{hash_s}$ using Algorithm 12
if $\mathbf{hash_s} == \mathbf{hash_d}$ **then**
  return Accept
else
  return Reject
end

1.9 Parameters settings

In this section, we propose our parameters aiming at three security levels in the new call of NISTPQC project [33] levels I, III and V, respectively.

1.9.1 List of our parameters

The key-size and the length of the signature are shown as below. Herein, the notation Size_{pk} denotes the public key size and Size_{sig} denotes the signature size. Note that the 16 Bytes **salt** is also indicated in the size of SNOVA signature (in order to attain EUF-CMA) and 16 Bytes seed length is included in the size of public key size.

Table 4: Table of key-sizes and lengths of the signature of SNOVA parameter settings (in bytes).

SL	(v, o, q, l)	Size_{pk}	Size_{sig}	$\text{Size}_{\text{esk}}/\text{Size}_{\text{ssk}}$
I	$(28, 17, 16, 2)$	9826(+16)	90(+16)	60008(+48)/48
	$(25, 8, 16, 3)$	2304(+16)	148.5(+16)	37962(+48)/48
	$(24, 5, 16, 4)$	1000(+16)	232(+16)	34112(+48)/48
III	$(43, 25, 16, 2)$	31250(+16)	136(+16)	202132(+48)/48
	$(49, 11, 16, 3)$	5989.5(+16)	270(+16)	174798(+48)/48
	$(37, 8, 16, 4)$	4096(+16)	360(+16)	128384(+48)/48
V	$(61, 33, 16, 2)$	71874(+16)	188(+16)	515360(+48)/48
	$(66, 15, 16, 3)$	15187.5(+16)	364.5(+16)	432297(+48)/48
	$(60, 10, 16, 4)$	8000(+16)	560(+16)	389312(+48)/48

1.9.2 How the performance are affected by parameters

The size of main term in public key is

$$\frac{m \cdot m^2 \cdot l^2 \cdot \log_2 q}{8}.$$

We can see the size of public key is mainly related to m, the number of ring equations, which is also the number of ring oil variables, the parameter o. Note that, the lower bound of m to achieve fixed security level is mainly determined by the attack in Sec. 4.3.2 and the number of vinegar variables is mainly determined by the analysis in Sec. 4.4.1. Therefore, for a fixed security level, by increasing the parameter l, we can further reduce value of m. Therefore, for larger l, we will have smaller public key size. On the other hand, the larger the l will make the signature size larger. It is a trade-off between small public key or small signature. Also, for larger l, the key generation will be more efficient, but the signing and verification will be less efficient, while still practical. We propose three parameter settings that allow the selection of a range of possible size and performance trade-offs. For the sake of security, we have chosen very conservative parameters.

2 Performance analysis (2.B.2)

2.1 Platforms used in the estimation

The benchmarking results of implementations are shown in Table 5. The reported values are the median cycles of 1000 executions compiled with "`gcc -O3`" with version

10.2.1. Our benchmarking is operated on a PC with an Intel Core i7-6700 CPU @ 3.40 GHz.

Note that, our optimized implementations only focus on multiplication of matrices without AVX2 hardware acceleration. We can expect faster implementation with AVX2 acceleration.

2.2 Time

Table 5: Benchmarking result of implementations (in CPU cycles).

SL	Schemes	KeyGen	Sign	Verify
I	$(28, 17, 16, 2)$-ssk	9595012	10964945	3161199
	$(28, 17, 16, 2)$-esk	9515984	4387586	3161199
	$(25, 8, 16, 3)$-ssk	3232587	12408096	3959869
	$(25, 8, 16, 3)$-esk	3232636	9825797	3959869
	$(24, 5, 16, 4)$-ssk	2441908	19681409	8086815
	$(24, 5, 16, 4)$-esk	2447831	17538089	8086815
III	$(43, 25, 16, 2)$-ssk	44658255	47587816	9443639
	$(43, 25, 16, 2)$-esk	44767410	15031152	9443639
	$(49, 11, 16, 3)$-ssk	23236343	60561733	18853861
	$(49, 11, 16, 3)$-esk	23264092	41138270	18853861
	$(37, 8, 16, 4)$-ssk	15295383	81382976	31084401
	$(37, 8, 16, 4)$-esk	15315591	68739319	31084401
V	$(61, 33, 16, 2)$-ssk	166196633	158443732	25289616
	$(61, 33, 16, 2)$-esk	166462440	36468696	25289616
	$(66, 15, 16, 3)$-ssk	81275417	172139775	47936266
	$(66, 15, 16, 3)$-esk	81451921	104229173	47936266
	$(60, 10, 16, 4)$-ssk	59318482	237150613	90472932
	$(60, 10, 16, 4)$-esk	59367845	186358881	90472932

2.3 Space

We summarize the estimations of key sizes of SNOVA scheme as follows.

Public key size. The reduced size of the public key of SNOVA using alignment is

$$\text{Size}_{\text{pk}} = \frac{m \cdot m^2 \cdot l^2 \cdot \log_2 q}{8} + |\mathbf{s_{public}}| = \frac{m \cdot m^2 \cdot l^2 \cdot \log_2 q}{8} + 16$$

bytes.

Expanded private key size. The size of private key is

$$\begin{aligned}\text{Size}_{\text{esk}} &= \frac{(m(n^2-m^2)l^2 + l^2vo + 4l^4)\cdot \log_2 q}{8} + |\mathbf{s_{public}}| + |\mathbf{s_{private}}| \\ &= \frac{(m(n^2-m^2)l^2 + l^2vo + 4l^4)\cdot \log_2 q}{8} + 48\end{aligned}$$

bytes.

Seed-type private key size. The size of the compressed private key is

$$\text{Size}_{\text{ssk}} = |\mathbf{s_{public}}| + |\mathbf{s_{private}}| = 32 + 16 = 48$$

bytes.

Signature size. The size of a signature of SNOVA scheme is

$$\text{Size}_{\text{sig}} = \frac{n\cdot l^2\cdot \log_2 q}{8} + |\mathbf{salt}| = \frac{n\cdot l^2\cdot \log_2 q}{8} + 16$$

bytes.

3 Expected Security Strength (2.B.4)

We give the expected security strength of our parameters aiming at three security levels in the new call of NIST PQC project [33] levels I, III and V, respectively.

3.1 Security strength

The following table shows the complexity of respective attacks against our parameters. "Dir.", "K-S.", "Int.", "$[T^{-1}]$.", "MinRank." and "Col." denote direct attack in Sec. 4.3.1, K-S attack in Sec. 4.4.3, intersection attack in Sec. 4.4.4 and equivalent key attack in Sec. 4.4.5, the complexity for the MinRank problem mentioned in Sec. 4.4.1 and the collision attack in Sec. 4.3.2, respectively.

In any pair of complexity the left one denotes the complexity in classical gates and the right one denotes in quantum gates, respectively. The lowest complexity is marked in bold fonts.

Table 6: Table of complexity in $\log_2(\sharp\text{gates})$.

SL	(v, o, q, l)	Dir.	K-S.	Int.	$[T^{-1}]$.	MinRank.	Col.
	$(28, 17, 16, 2)$	171/124	181/93	275	192/192	**151**	**151**
I	$(25, 8, 16, 3)$	175/126	617/311	819	231/231	**148**	159
	$(24, 5, 16, 4)$	188/134	1221/613	1439	286/286	**150**	175
	$(43, 25, 16, 2)$	240/175	293/149	439	279/ 279	**212**	215
III	$(49, 11, 16, 3)$	230/162	1373/689	1631	530/530	215	**213**
	$(37, 8, 16, 4)$	291/214	1861/933	2192	424/424	**217**	271
	$(61, 33, 16, 2)$	308/224	453/229	727	386/386	**279**	**279**
V	$(66, 15, 16, 3)$	307/220	1841/923	2178	707/707	**280**	285
	$(60, 10, 16, 4)$	355/255	3205/1605	3602	812/812	**278**	335

4 Analysis With Respect To Known Attacks (2.B.5)

To start with, when working with an equation over matrix ring $\mathcal{R}$, we can observe that the components will give us l^2 equations over the ring variables' entry elements in $\mathbb{F}_q$. For instance, when $l = 2$, we consider $A, B \in \mathcal{R}$ and ring variables X, Y. The components on both sides of the ring equation

$$\begin{bmatrix} x_1 & x_3 \\ x_2 & x_4 \end{bmatrix} \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} y_1 & y_2 \\ y_3 & y_4 \end{bmatrix} = X^t A Y = B = \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix}$$

give us $4 = 2^2$ quadratic equations over $\mathbb{F}_q$.

Therefore, SNOVA can be considered both a UOV-like signature scheme over the matrix ring $\mathcal{R}$ and a UOV over $\mathbb{F}_q$. The security analysis are presented from two different aspects: over the ring $\mathcal{R}$ and over the finite field $\mathbb{F}_q$.

4.1 A note of our analysis

The target of this section is to explore various methods of attacking the SNOVA and assess their feasibility. The key observation is that when a signature scheme is based on a quadratic form over ring (as in the case of SNOVA), its private key T is shared with another signature scheme over ring whose structure is much simpler. To conduct a comprehensive and prudent security analysis, we start with the following notions.

Ring UOV. Notice that the central map of SNOVA is of the form, $i = 1, \ldots, m$,

$$F_i(X_1, \ldots, X_n) = \sum_{\alpha=1}^{l^2} A_\alpha \cdot \left(\sum_{(j,k)\in\Omega} X_j^t \left(Q_{\alpha 1} F_{i,jk} Q_{\alpha 2}\right) X_k \right) \cdot B_\alpha$$

and the public key is generated via the congruence relation $[P_i] = [T]^t [F_i] [T], i = 1, \ldots, m$. Therefore, we can construct a UOV scheme over $\mathcal{R}$ that shares the same private key T with SNOVA scheme. Namely, the UOV scheme over $\mathcal{R}$ whose central map has the form, $i = 1, \ldots, m$,

$$\tilde{F}_i(X_1, \ldots, X_n) = \sum_{(j,k)\in\Omega} X_j^t F_{i,jk} X_k$$

and the corresponding public key $\tilde{P} = \left[\tilde{P}_1, \cdots, \tilde{P}_m\right]$ where

$$\tilde{P}_i = \tilde{F}_i \circ T$$

and T is private key of SNOVA. Then, if an attacker recovers the private key T by attacking this ring UOV over $\mathcal{R}$ then he also find an equivalent key of SNOVA scheme since both schemes generate their public key through the same congruence relation $[T]^t [F_i] [T]$.

Key recovery attacks. As we already seen, a (v, o, q, l) SNOVA scheme shares its private key T with the corresponding (v, o, q, l) ring UOV scheme. Therefore, for key recovery attacks, the security of SNOVA is evaluated by analyzing the complexity of such attacks against the associated ring UOV scheme.

To our best knowledge, we do not find a complete key recovery attack against this ring UOV. On the other hand, this ring UOV still induces a UOV over field. Some complexity estimations of the related problems of this ring UOV and its corresponding UOV over field will be discussed in 4.4.1. Attacks against some UOV-like schemes will also be mentioned in 4.4.1. We will analyze the structure utilized in these attacks as well as their main insight, and then discuss their potential feasibility for attacking SNOVA.

Forgery attacks. Finding the preimage of the public map for the hash value of a message is what constitutes signature forgery. However, the public maps of SNOVA and ring UOV are only weakly connected as a result of the use of l^2 copies with different A_α, $Q_{\alpha 1}$, $Q_{\alpha 2}$, and B_α in F_i of SNOVA. Consequently, solving the equations derived from the public map of ring UOV does not aid in solving the equations produced by the public map of SNOVA for the purpose of forgery attacks. Therefore, the security of forgery attacks will be analyzed with respect to the public key of the SNOVA scheme.

Except these, one may directly forge valid fake signature of SNOVA over $\mathcal{R}$ not returning to field level. This approach will suffer from the fact that there is no efficient algorithm like F_4, F_5 and XL to solve multivariate quadratic system over the noncommutative ring $\mathcal{R}$. Based on this fact, we will return to the field level and estimate the associated complexity.

For our parameter sets, the critical attacks are direct attack 4.3.1, collision attack 4.3.2, MinRank attack 4.4.1 and equivalent key attack 4.4.5. The complexity estimations of K-S attack 4.4.3 and intersection attack 4.4.4 show that they are not the most crucial to our parameter sets.

4.2 Preliminaries

In this section, we briefly describe the tools that we used to estimate the complexity of solving a MQ problem and a MinRank problem, respectively.

4.2.1 Solving MQ systems and complexity estimation

There are several algorithms to solve a quadratic system of M equations in N variables over finite fields such as F_4 [20], F_5 [21] and XL variants [15, 12, 52].

Solving MQ problem. The complexity of solving M homogeneous quadratic equations in N variables [6, 12] can be estimated by

$$MQ(N, M, q) = 3 \cdot \binom{N - 1 + d_{reg}}{d_{reg}}^2 \cdot \binom{N+1}{2}$$

field multiplications where d_{reg} is the degree of regularity of a semi-regular polynomial system and it is equal to the smallest positive integer such that the coefficient of t^d term in the series generated by

$$\frac{(1-t^2)^M}{(1-t)^N}$$

is non-positive.

Hybrid approach. The hybrid approach [5] randomly guesses k variables before solving the MQ system and the corresponding complexity is $q^k \cdot MQ(N-k+1, M, q)$ field multiplications for the classical case and $q^{k/2} \cdot MQ(N-k+1, M, q)$ field multiplications when applying Grover's algorithm [24] for the quantum case.

Methods solving underdetermined MQ. On the other hand, Thomae and Wolf [46], Furue, Nakamura and Takagi [22], Hashimoto [25] provide several methods to solve an underdetermined multivariate quadratic system P of M equations in N variables over a finite field, i.e., $N > M$. The main idea is to find a particular invertible linear map S converting the first α_k equations into a special form where k is the number of guessing in the hybrid approach. We can then remove $(N - M) + \alpha_k$ variables and α_k equations from system P.

Therefore, an underdetermined $MQ(N,\ M,\ q)$ problem reduces to an $MQ(M-k-\alpha_k+1,\ M-\alpha_k,\ q)$ problem and hence can by solved using the hybrid approach [5]. Note that different methods obtain different optimal values α_k due to how they convert P into different forms. Therefore, the formulas for estimation of complexity of [46, 22, 25] are the same but with different optimal values α_k. Hence, the main term of complexity of solving MQ system under this technique is given by

$$\min_k\ q^k \cdot MQ(M - k - \alpha_k + 1,\ M - \alpha_k,\ q)$$

field multiplications in the classical case and

$$\min_k \; q^{k/2} \cdot MQ(M-k-\alpha_k+1,\; M-\alpha_k,\; q)$$

in the quantum case with different optimal values α_k corresponding to different methods.

The optimal values α_k of [46, 22] are $\alpha_{\mathrm{TW}} = \lfloor \frac{N}{M} \rfloor - 1$, $\alpha_{\mathrm{F}} = \lfloor \frac{N-k}{M-k} \rfloor - 1$, respectively, and $\alpha_{\mathrm{HMa}} = \lfloor \frac{N}{M-k} \rfloor - 1$, α_{HMb} is the maximal integer such that $N \geq M - (\alpha_k + k - M)\alpha_k$ holds, where α_{HMa} and α_{HMb} are corresponding to the two algorithms proposed in [25], respectively. Note that, the estimation of the direct attack in Sec. 4.3.1 would be the sharpest one among [46, 22, 25].

Algorithms for super-underdetermined MQ. Note that, [28, 14, 31, 13] indicate that when the number of variables N is sufficiently larger than the number of equations M in a MQ problem then we can solve this MQ in polynomial time. Please refer to the table in [25] for more information. Note that these four algorithms are not applicable to the parameter settings of SNOVA.

4.2.2 MinRank problem and Support-Minors modeling

MinRank problem. For $M_1, \cdots, M_k \in \mathbb{F}_q^{M \times N}$ and a target rank r, the MinRank problem asks to find a non-trivial linear combination of the matrices which has rank at most r. That is, to find a vector $\vec{\mathbf{x}} = (x_1, \cdots, x_k)^t \in \mathbb{F}_q^k$ such that

$$\mathrm{rank}\left(\sum_{i=1}^{k} x_i M_i\right) \leq r.$$

Solving MinRank problem. Notice that the MinRank problem is NP-hard [11] and it plays a central role in the cryptanalysis of MPKC. Recently, Bardet *et al.* proposed the Support-Minors (SM) modeling algorithm [3] to solve MinRank problem. This powerful algorithm transform the rank condition into a large bilinear system which is sparse and then use the linearization method to solve it. The complexity of this algorithm is estimated by

$$MinRank(M,N,k,r) = 3 \cdot k(r+1) \cdot \left(\binom{N}{r} \binom{k+b-1}{b} \right)^2$$

where b is the smallest positive integer such that

$$\binom{N}{r}\binom{k+b-1}{b} - 1 \leq \sum_{i=1}^{b} (-1)^{i+1} \binom{N}{r+i} \binom{M+i-1}{i} \binom{k+b-i-1}{b-i}$$

holds.

Moreover, Bardet *et al.* point out that one may choose to use the first $N' \leq N$ columns when applying their algorithm and for some optimal N' so that $r+1 \leq N' \leq N$ the

cost of computation can be further reduced. Our security analysis will also consider this technique in our estimations.

Superdetermined MinRank problem. Superdetermined MinRank problem is defined in [47] as the MinRank problem with $k < rM$. Moreover, in [1] Bardet and Bertin indicate that the modeling in [47] can be seen as a special case of SM modeling and the best complexity will be the one that solving the associated Macaulay matrix by linearization. If we consider the minors as new variables, the system can be solved whenever $M(N-r) \geq k(r+1)$, i.e., $b = 1$. Moreover, with Plücker coordinates, the Macaulay matrix has a special form and this can help us to solve the problem more quickly. For $1 \leq d \leq r-1$, if

$$m\binom{n-r}{d+1}\binom{r}{d} \geq k\binom{n-r}{d+1}\binom{r}{d+1} + k\binom{n-r}{d}\binom{r}{d} - 1,$$

then with overwhelming probability the solution can be obtained [1].

SM modeling with hybrid technique. In [2], Bardet *et al.* show that we can solve a $MinRank(M, N, k, r)$ problem by performing $q^{\alpha r}$ attacks on those much more smaller $MinRank(M, N-\alpha, k-\alpha m, r)$ instances where α is a positive integer so that $k - \alpha m \geq 0$ and then only one of them has the solution. Therefore, the complexity of SM modeling with hybrid technique is

$$\min_{\alpha \geq 0} q^{\alpha r} \cdot MinRank(M, N-\alpha, k-\alpha m, r).$$

4.3 Forgery attacks

4.3.1 Direct attack

For a quadratic multivariate polynomial system $P = [P_1, \cdots, P_m]$ consisting of m equations in n variables over $\mathbb{F}_q$ and $\vec{\mathbf{y}} \in \mathbb{F}_q^m$, an attacker can directly try to solve the solution $\vec{\mathbf{u}}$ of the system $P(\vec{\mathbf{u}}) = \vec{\mathbf{y}}$ algebraically with Gröbner basis approach such as [20, 21, 15, 12, 52]. In the case of UOV ($n > m$), the public key is underdetermined, the methods of solving underdetermined MQ are applicable. Therefore we can assign the values to $n-m$ variables in the system $P(\vec{\mathbf{u}}) = \vec{\mathbf{y}} = Hash(\mathbf{digest}||\mathbf{salt})$ randomly and then obtain a MQ system of m equations in m variables which can be solved with high probability. Once the system can be solved, the solution $\vec{\mathbf{u}}$ will be a valid fake signature and hence $P(\vec{\mathbf{u}}) = \vec{\mathbf{y}}$.

In the case of SNOVA, to produce a fake signature, an attacker need to regard a (v, o, q, l) SNOVA as a (l^2v, l^2o, q) UOV and then forge a signature for this UOV. Each equation over $\mathcal{R}$ yields l^2 equations over $\mathbb{F}_q$ and then the system over ring $\mathcal{R}$, $P(\vec{\mathbf{U}}) = \vec{\mathbf{Y}}$, will result in an MQ system consisting of l^2m equations in l^2m field variables.

Table 7 gives comparison of the degree at the first step degree falls or goes flat using F_4 algorithm [20], which is strongly connected to the degree of regularity [18], in Magma

algebra system [9] that starts to go either down or flat among all step degrees of the quadratic system obtained by SNOVA and a random quadratic system respectively.

In random systems, the first fall step degree is generally equal to the degree of regularity. Table 7 indicates that the first fall step degrees of SNOVA systems and random systems are identical for small size parameter sets. Thus, we can expect that the degree of regularity of SNOVA systems, the first fall step degree, and the degree of regularity of random systems are the same. For Gröbner bases algorithms such as F4/5 and XL, the size of the Macaulay matrix employed in solving quadratic systems is determined by the degree of regularity. The complexity of solving quadratic systems is determined by the difficulty of solving the sparse Macaulay matrix using the Wiedemann solver [53]. As a result, the complexity of a direct attack on SNOVA is estimated by the complexity of a direct attack on random systems.

The complexity estimations of classical direct attack are

$$\text{Comp}_{\text{Direct; Classical}}^{\text{SNOVA}} = \min_k \; q^k \cdot MQ(l^2m - k - \alpha_k + 1,\; l^2m - \alpha_k,\; q)$$

and with Grover's algorithm [24].

$$\text{Comp}_{\text{Direct; Quantum}}^{\text{SNOVA}} = \min_k \; q^{k/2} \cdot MQ(l^2m - k - \alpha_k + 1,\; l^2m - \alpha_k,\; q)$$

field multiplications where α_k is obtained by the technique in [46, 22, 25].

Table 7: Table of comparison of the degree at the first step degree falls or goes flat between SNOVA and random systems. Our experiment shows that in the case of small size parameter sets such a quadratic system over field induced by SNOVA public key behaves like a random systems of $l^2 \cdot m$ equations in $l^2 \cdot m$ variables over a $\mathbb{F}_q$.

(v, o, q, l, k)	SNOVA system	random system
$(6, 1, 16, 2, 1)$	3	3
$(6, 2, 16, 2, 1)$	5	5
$(6, 2, 16, 2, 2)$	4	4
$(6, 2, 16, 2, 3)$	3	3
$(6, 3, 16, 2, 1)$	7	7
$(6, 3, 16, 2, 2)$	6	6
$(6, 3, 16, 2, 3)$	5	5
$(6, 4, 16, 2, 2)$	7	7
$(6, 4, 16, 2, 3)$	6	6
$(6, 1, 16, 3, 2)$	4	4
$(6, 1, 16, 3, 3)$	4	4
$(6, 1, 16, 3, 4)$	3	3
$(6, 2, 16, 3, 3)$	7	7
$(6, 2, 16, 3, 4)$	6	6
$(6, 2, 16, 3, 5)$	5	5
$(6, 1, 16, 4, 1)$	9	9
$(6, 1, 16, 4, 2)$	7	7
$(6, 1, 16, 4, 3)$	6	6
$(6, 1, 16, 4, 4)$	5	5
$(6, 1, 16, 4, 5)$	5	5

4.3.2 Collision attack

To forge a fake signature, an attacker can try to obtain the values M signatures $\overrightarrow{\mathbf{U}_j}$ where $j = 1, \cdots, M$ and N hash values $Hash(\mathbf{digest}||\mathbf{salt}_k)$ where $k = 1, \cdots, N$, if there exists a collision $P(\overrightarrow{\mathbf{U}_j}) = Hash(\mathbf{digest}||\mathbf{salt}_k)$ then it would be a valid fake signature.

Thus, M signature computations and N hash values computations are involved. Therefore, according to the estimation of [10], the cost of collision attack would be

$$M \cdot (l^2 m) \cdot \left(2(\log_2 q)^2 + 3 \cdot \log_2 q\right) + N \cdot 2^{17}$$

gates in the sense that regarding SNOVA as a UOV scheme.

Note that the lower bound of the complexity of collision attack is

$$2 \cdot \left(M(l^2m)\left(2(\log_2 q)^2 + 3 \cdot \log_2 q\right) \cdot N2^{17}\right)^{1/2}$$

gates.

If $MN = q^{l^2m}$, then this lower bound turns into

$$2 \cdot \left(q^{l^2m}(l^2m)\left(2(\log_2 q)^2 + 3 \cdot \log_2 q\right) \cdot 2^{17}\right)^{1/2},$$

and the collision exists with probability

$$\begin{aligned} 1 - \left(\frac{q^{l^2m} - M}{q^{l^2m}}\right)^N &= 1 - \left(\frac{MN - M}{MN}\right)^N \\ &= 1 - \left(1 - \frac{M}{MN}\right)^N \\ &\approx 1 - e^{\left(\frac{1}{N}\right)N} \\ &= 1 - e^{-1}. \end{aligned}$$

4.4 Key Recovery Attacks

4.4.1 Quadratic forms over ring and MinRank attacks

In this subsection, we will analyze the structure of ring UOV corresponding to SNOVA and estimate the complexity when solving related MinRank problem.

Kernel of ring UOV. The central map F of UOV is an multivariate quadratic map vanished on the oil (linear) space $\mathcal{O}$. We discover that if we regard a (v, o, q, l) ring UOV as a (l^2v, l^2o, q) UOV scheme over $\mathbb{F}_q$, then the matrices corresponding to central map of this UOV (induced by the ring UOV), say $M_1, \cdots, M_{l^2o}$, are sparse.

We can see that for each $1 \leq i \leq l^2o$, the matrix M_i vanishes on a linear space $\mathcal{W}_i$ such that $\mathcal{O} \subseteq \mathcal{W}_i$ and $\dim \mathcal{O} \leq \dim \mathcal{W}_i$. However, the intersection of $\mathcal{W}_i$ is still the oil space $\mathcal{O}$ (we can easily see this phenomenon in toy examples).

Therefore, we conclude that this (l^2v, l^2o, q) UOV will not vanish on a linear space which is larger than the oil space $\mathcal{O}$. This observation provides us with some confidence in the security of this UOV against traditional key recovery attacks, such as the Kipnis-Shamir attack [28], reconciliation attack [19], and intersection attack [7] in the sense that this UOV has an oil-vinegar structure similar to the original UOV.

No multi-layer structure. One may worry that this sparsity will lead to some structures that are known to be broken, such as, multi-layer structure. In [7, 6], Beullens

proposed a series of MinRank attack against Rainbow [17] scheme based on its multi-layer structure. Such multi-layer structure will result in nested structure of oil spaces [7] and the low-rankness can be used to find a vector in the linear space $T^{-1}(\mathcal{O})$ and hence an equivalent key.

Another attack that uses multi-layer structure is the MinRank attack proposed by Thomae [45] against NC-Rainbow [56] which is a variant of Rainbow which is based on Quaternion ring over a finite field $\mathbb{F}_q$ of characteristic 2. If an attacker regards an NC-Rainbow scheme as a Rainbow scheme over $\mathbb{F}_q$, then the rank of the corresponding matrix to the central map F of NC-Rainbow will be lower than original Rainbow. This low-rankness of central map matrices comes from its sparse form caused by the special structure of multiplication of Quaternion ring. Then, this low-rankness makes the MinRank attacks that attacking multi-layer structure more efficient.

Although some of matrices among $M_1, \cdots, M_{l^2o}$ may have rank lower than the others, there is no multi-layer structure in their relation. That is, SNOVA has no multi-layer structure, we can not regard a SNOVA scheme (and similarly the corresponding ring UOV scheme) as a Rainbow scheme.

Consequently, attacks [7, 6, 45] that rely on the multi-layer structure have no security impact on the UOV induced by the quadratic form over ring and thus will not affect the security of SNOVA.

Intersection of the null spaces of public key differential. In [35], Park broke the Matrix-based UOV scheme [44] which is proposed by Tan and Tang.

The main insight of this attack is: when an attacker regards the matrices of the differential of the central map and the public key of Matrix-based UOV as linear operators, the sparsity of some of these matrices makes the intersections of the corresponding null spaces non-trivial, while general UOV do not have this phenomenon. Park also showed that any basis of this non-trivial intersection can be used to build an equivalent private key.

Note that the null spaces of the differential of the central map and the public key of the ring UOV corresponding to SNOVA have no structure same as that in Matrix-based UOV. Therefore, the attack in [35] is not applicable to this ring UOV and hence the attack will not affect the security of SNOVA.

Matrices may have low rank. When we regard this (v, o, q, l) ring UOV as a (l^2v, l^2o, q) UOV scheme over $\mathbb{F}_q$, we discover that some corresponding matrices has rank at most lv. To our best knowledge, we do not find a complete key recovery attack against Ring UOV scheme based on this MinRank problem.

However, this phenomenon still induces a $MinRank(l^2n, l^2n, l^2m, lv)$ MinRank problem. For the sake of security, we estimate the complexity of solving this MinRank problem using Support-Minors algorithm and take this into account when we choose our parameter settings of SNOVA scheme.

$$\text{Comp}_{\text{MinRank}}\text{SNOVA} = MinRank(l^2n, l^2n, l^2m, lv).$$

If there were a key recovery attack using this MinRank problem, then its complexity should be greater than this MinRank problem. Hence the security of our parameter setting will not be affected.

Superdetermined and hybrid approach. The $MinRank(l^2n, l^2n, l^2m, lv)$ instance above is superdetermined and then the technique in [1] can be applied to this instance. Note that [1] also shows that executing the computation at the smallest degree and with the smallest number of variables will not always be the best estimation. In conclusion, our complexity estimations take both strategies, the technique in [1] and solving system in higher degree $b > 1$ [3], into consideration. As a result, the approach in [1] will not affect the security of our parameters.

On the other hand, note that the hybrid approach in [2] is not applicable to the instance above since the MinRank problem is required to be underdetermined. Therefore, these two approaches are not crucial for our parameters.

4.4.2 Reconciliation Attack

The reconciliation attack proposed by [19] against UOV is trying to find a vector $\vec{\mathbf{o}} \in T^{-1}(\mathcal{O})$ by solving the system $P(\vec{\mathbf{o}}) = 0$ and hence the basis of $T^{-1}(\mathcal{O})$ can be recovered. This implies that $P(\vec{\mathbf{o}}) = 0$ is a quadratic system that having a solution space of dimension m. To expect a unique solution, we can impose m linear constraints with respect to the components of $\vec{\mathbf{o}}$. Hence the complexity of this attack is mainly given by that of solving the quadratic system of m equations in v variables.

A reconciliation attack on SNOVA, if considered over field, is as an attack on UOV, thus we are in the case of solving the quadratic system of l^2m equations in $v > o = l^2m$ variables. Hence the reconciliation attack usually will not outperform the direct attack on the public key of SNOVA in which the complexity comes from solving l^2m quadratic equations in l^2m variables. Furthermore, the direct attack on the public key will appear to be more efficient due to the technique that solving underdetermined quadratic equations 4.3.1.

4.4.3 Kipnis-Shamir attack (UOV attack)

The K-S attack [28] is trying to find an equivalent private key by finding an equivalent invertible linear map T and hence the corresponding matrix $[T]$. Once we have an equivalent $[T]$, we can recover equivalent $[F_i]$ by the relation $[F_i] = [T^{-1}]^t [P_i] [T^{-1}]$. Note that [28] shows that $T^{-1}(\mathcal{O})$, the oil subspace of the public key P of UOV, induces an equivalent key.

In [28, 7], it shows that $T^{-1}(\mathcal{O})$ is an invariant subspace of $[P_i']^{-1}\left[P_j'\right]$. The K-S attack is trying to find a vector in $T^{-1}(\mathcal{O})$. Once one such vector is found, then we expect that the whole space $T^{-1}(\mathcal{O})$ can be recovered efficiently by using method in [7]. A vector in $T^{-1}(\mathcal{O})$ can be expected to be found with q^{n-2m} attempts. Note that if there are $[P_i']$'s not invertible, then we can replace $[P_i']$ with invertible linear combinations of $[P_i']$'s randomly chosen and the cryptanalysis of K-S attack remains the same.

Therefore the complexities of K-S attack and quantum K-S attack are

$$\text{Comp}_{\text{K-S; classical}}\text{UOV} = q^{n-2m-1}$$

field multiplications and

$$\text{Comp}_{\text{K-S; quantum}}\text{UOV} = q^{(n-2m-1)/2}$$

field multiplications, respectively.

From the design of central map F of SNOVA and the noncommutativity of $\mathcal{R}$, there does not exist the notion of oil space of F over $\mathcal{R}$ analogous to the space $\mathcal{O}$ of UOV and hence the notion of $T^{-1}(\mathcal{O})$ in the sense that regarding $T^{-1}(\mathcal{O})$ as a left-module or a right-module over $\mathcal{R}$. Such a requirement is necessary for K-S attack, since to execute K-S attack over $\mathcal{R}$, the consistency of multiplication over $\mathcal{R}$ given by a left-module or a right-module over $\mathcal{R}$ is needed. Therefore, K-S attack is not applicable to SNOVA over $\mathcal{R}$. Note that [36] also proposes two methods to find an invariant subspace: the Linearization method and the Characteristic Polynomial method. These two methods become invalid over $\mathcal{R}$ since they still suffer from the noncommutativity of $\mathcal{R}$.

However, an attacker may treat the ring UOV which is corresponding to the (v, o, q, l) SNOVA scheme over $\mathcal{R}$ as an (l^2v, l^2o, q) UOV system over $\mathbb{F}_q$ and then carry out the K-S attack over $\mathbb{F}_q$.

Then we have

$$\text{Comp}_{\text{K-S; classical}}\text{SNOVA} = q^{l^2n-2l^2m-1}$$

field multiplications for classical attack and

$$\text{Comp}_{\text{K-S; quantum}}\text{SNOVA} = q^{(l^2n-2l^2m-1)/2}$$

field multiplications for quantum attack.

4.4.4 Intersection attack

In [7], Beullens proposed the intersection attack to attack UOV scheme. It uses the polar form of the public key P, that is, $P' = [P_1', \cdots, P_m']$ with $P_i'(\overrightarrow{\mathbf{u_1}}, \overrightarrow{\mathbf{u_2}}) = \overrightarrow{\mathbf{u_1}}^t [P_i'] \overrightarrow{\mathbf{u_2}}$ where $[P_i'] = [P_i] + [P_i]^t$.

The intersection attack is trying to first find a vector $\vec{\mathbf{y}}$ in the subspace, namely the intersection $\Big([P_i'] \, (T^{-1}\mathcal{O}) \Big) \cap \Big(\left[P_j'\right] \, (T^{-1}\mathcal{O}) \Big)$ where $[P_i'] \, , \left[P_j'\right]$ are invertible, and then to obtain an equivalent key by recovering the subspace $T^{-1}(\mathcal{O})$.

Since $([P_i']^{-1})\vec{\mathbf{y}}$, $(\left[P_j'\right]^{-1})\vec{\mathbf{y}} \in T^{-1}(\mathcal{O})$, we obtain the following system.

$$\begin{cases} P\Big(\left([P_i']^{-1}\right) \vec{\mathbf{y}} \Big) = 0 \\ P\Big((\left[P_j'\right]^{-1}) \vec{\mathbf{y}} \Big) = 0 \\ P'\Big(([P_i']^{-1}) \vec{\mathbf{y}}, (\left[P_j'\right]^{-1}) \vec{\mathbf{y}} \Big) = 0 \end{cases}$$

Whenever $\mathbf{2.5m < n < 3m}$. If $2.5m < n < 3m$, there is a $3m - n$ dimensional subspace of solutions. To obtain a unique solution with high probability, we can add $3m-n$ linear random equations. Hence the complexity of solving the system is equivalent to that of solving quadratic system with $M = 3m$ equations and $N = n - (3m - n) = 2n - 3m$ variables. Then the complexity is

$$\text{Comp}_{\text{Intersection}}\text{UOV} = MQ(N+1,\ M,\ q)$$

field multiplications.

Whenever $\mathbf{n < 2.5m}$. If $n < 2.5m$, the attack can become more powerful by seeking a vector $\vec{\mathbf{y}}$ in the intersection of k subspaces $[P_i']^{-1} \, (T^{-1}\mathcal{O})$ with $k \geq 2$. The complexity of this case is equal to the complexity of that solving the quadratic system with $M = \binom{k+1}{2}m - 2\binom{k}{2}$ equations and $N = nk - (2k-1)m$ variables.

Therefore, when $n < 2.5m$, we have $N = nk - (2k-1)m,\ M = \binom{k+1}{2}m - 2\binom{k}{2}$, and

$$\text{Comp}_{\text{Intersection}}\text{UOV} = MQ(N+1,\ M,\ q)$$

field multiplications.

In case of intersection attack against SNOVA, due to our construction, we can not write the public polynomial P_i of SNOVA in quadratic form, namely $\overrightarrow{\mathbf{u_1}}^t \, [P_i'] \, \overrightarrow{\mathbf{u_2}}$, when considered as over $\mathcal{R}$. Thus, the implementation of intersection attack still face the same problem as in direct attack, that is, there is no efficient algorithm like F_4, F_5 and XL to compute.

From our perspective, to implement intersection attack against SNOVA, an alternative suitable strategy is regarding the ring UOV corresponding to SNOVA as a UOV system over $\mathbb{F}_q$ and then solve a system over $\mathbb{F}_q$. Therefore, the complexity is estimated by the following

Whenever $\mathbf{n < 2.5m}$. If $n < 2.5m$, we have $N = (l^2 n)k - (2k-1)(l^2 m),\ M = \binom{k+1}{2}(l^2 m) - 2\binom{k}{2}$, and

$$\text{Comp}_{\text{Intersection}}\text{SNOVA} = MQ(N+1,\ M,\ q)$$

field multiplications.

Whenever 2.5m < n < 3m. In the case $2.5m < n < 3m$, $N = 2(l^2n) - 3(l^2m),\ M = 3(l^2m)$, and

$$\text{Comp}_{\text{Intersection}}\text{SNOVA} = MQ(N+1,\ M,\ q)$$

field multiplications.

Whenever n ≥ 3m. If $n \geq 3m$, then there is no guarantee that the subspace, namely the intersection $\left([P_i'] \left(T^{-1}\mathcal{O}\right) \right) \cap \left([P_j'] \left(T^{-1}\mathcal{O}\right) \right)$ will exist. Therefore, the intersection attack becomes a probabilistic attack against SNOVA. In this case, the complexity is

$$\text{Comp}_{\text{Intersection}}\text{SNOVA} = q^{(l^2n)-3(l^2m)+1} \cdot MQ(N+1,\ M,\ q)$$

field multiplications where $N = l^2n, M = 3(l^2m)$.

Our experiment shows that the quadratic system induced by intersection attack on ring UOV will not degenerate, e.g., in the toy example $(v, o, q, l) = (3, 2, 16, 2)$, that is, behaves like a semi-regular system. We consider this result to be natural since the kernel of ring UOV is the oil space $T^{-1}(\mathcal{O})$ which is similar to the original UOV scheme as mentioned in 4.4.1 and this is the space that the intersection attack tries to detect.

4.4.5 Equivalent key attack

An attacker may try to find the submatrix $(T^{-1})^{12}$ of matrix $[T^{-1}]$ in the top right corner by algebraic attacks. Once the matrix $[T^{-1}]$ is found, the central map F can be recovered. This can be done by considering the system $P(T^{-1}(\vec{\mathbf{x}})) = F(\vec{\mathbf{x}})$ and solve for $[T^{-1}]$ by comparing both sides of equation at ring level. Then it induces a system of $m \cdot m^2 \cdot l^2$ quadratic equations in lvo variables over $\mathbb{F}_q$ and hence can be solved by F_4, F_5 and XL.

Therefore, the complexity is

$$\text{Comp}_{T^{-1}}\text{SNOVA} = MQ(lvo+1,\ m^3l^2,\ q)$$

field multiplications.

Note that the multivariate quadratic system constructed by this attack is overdetermined, hence [27, 14, 31, 13, 46, 22, 25] are not applicable.

On the other hand, one may consider that executing equivalent key attack that regards a (v, o, q, l) SNOVA as an (l^2v, l^2o, q) UOV then inducing a quadratic system of $M = (l^2m) \cdot (l^2m) \cdot \frac{l^2m+1}{2}$ equations in $N = lvo$ variables over $\mathbb{F}_q$. However, our experiments show that this formulation does not increase the number of independent equations.

With the table 8, we hope to provide some information and the trend of d_{reg} on this equivalent key attack when l increasing.

Table 8: Trend table of changes in degree of regularity.

(v, o, q, l)	N	M	d_{reg}
$(28, 17, 16, 2)$	952	19652	11
$(25, 8, 16, 3)$	600	4608	16
$(24, 5, 16, 4)$	480	2000	23
$(19, 6, 16, 4)$	456	3456	13

5 Advantages and limitations (2.B.6)

5.1 Advantages

The main advantages of SNOVA are as follows.

- **Small public key sizes and signature sizes:** As an MQ-problem based signature scheme, the signature sizes of SNOVA are tiny as usual. However, SNOVA also enjoys very small public key sizes. Even for NIST security level three, we could have a pair of public key size 4112 bytes and signature size 376 bytes.
- **Modest computational requirements:** During the signing and verification, we only need to do simple matrix operations over a finite field. Thus, it can be easily implemented on mobile devices.
- **Small secret key:** We may use two seeds together as our seed-type secret key which is as small as 48 bytes.
- **A wide security margin:** We are very conservative in our security analysis, thus leaving a wide security margin.
- **Simple arithmetic:** Although the core idea of SNOVA is the use of noncommutative ring, the underlying basic operations to achieve the goal are really linear algebra. Therefore, once understanding the nuance of the delicate design, the simplicity of the SNOVA is really almost UOV, plus noncommutativity.

It is worth mentioning that, the protocol TLS, which is used to protect our web browsing, will be no longer secure due to the impact of quantum computers as pointed out in [54, 55]. Making TLS post-quantum is an important task, but such a fundamental change could take years and be quite costly if we do not have a quantum-resistant signature that is relatively well compatible with the existing framework. In particular, [55] gives the corresponding condition: six times signature size and two times of public key size fit in 9KB. According to the specification of SNOVA, SNOVA could be a more practical general-purpose signature scheme.

5.2 Limitations

- **No provable security:** SNOVA, like all known MQ-based cryptosystems, has no provable security. However, if we take our coefficients in the noncommutative ring to be solely in the center of it, then SNOVA is reduced to a small UOV. Since UOV is a well-studied case, therefore we have strong confidence in the security of SNOVA.

- **Performance trade-off:** Unavoidably, there is a trade-off between public key size and performance. Under the premise of being conservative about security, the parameter sets we proposed are still practical.

- **Choices on l in $\mathrm{Mat}_{l\times l}(\mathbb{F}_q)$:** Our actual implementation shows that the l in $\mathrm{Mat}_{l\times l}(\mathbb{F}_q)$ will influence the size of public key and signatures. Keeping the same level of security, the bigger l will result in smaller public key size but larger signatures. However, larger l will give a draw back on performance. Currently, the parameter l are limited to 2, 3, and 4 in our design.

Also, to avoid possible potentially weak keys, we may require that A_α's, B_α's, $Q_{\alpha 1}$'s, $Q_{\alpha 2}$'s being invertible, although it is not a requirement for the scheme to work properly.

References

[1] Bardet, M., Bertin, M.: **Improvement of Algebraic Attacks for Solving Superdetermined MinRank Instances.** In: Cheon, J.H., Johansson, T. (eds) Post-Quantum Cryptography. PQCrypto 2022. Lecture Notes in Computer Science, vol 13512. Springer, Cham. https://doi.org/10.1007/978-3-031-17234-2_6

[2] Bardet, M., Briaud, P., Bros, M., Gaborit, P., Tillich, J.P.: **Revisiting Algebraic Attacks on MinRank and on the Rank Decoding Problem.** Available at https://eprint.iacr.org/2022/1031.pdf.

[3] Bardet, M., Bros, M., Cabarcas, D., Gaborit, P., Perlner, R.A., Smith-Tone, D, Tillich, J.P., Verbel, J.A.: **Improvements of algebraic attacks for solving the rank decoding and MinRank problems.** In Shiho Moriai and Huaxiong Wang, editors, ASIACRYPT 2020, Part I, volume 12491 of LNCS, pages 507–536. Springer, Heidelberg, December 2020.

[4] Bardet, M., Faugère, J. C., Salvy, B., Yang, B. Y.: **Asymptotic behavior of the index of regularity of quadratic semi-regular polynomial systems.** In 8th Interna- tional Symposium on Effective Methods in Algebraic Geometry (MEGA), pp. 1–14 (2005).

[5] Bettale, L., Faugère, J.-C., Perret, L.: **Hybrid approach for solving multivariate systems over finite fields.** Journal of Mathematical Cryptology 3, pp. 177–197 (2009).

[6] Beullens, W.: **Breaking Rainbow Takes a Weekend on a Laptop.** Cryptology ePrint Archive, Report 2022/214, 2022. https://eprint.iacr.org/2022/214.pdf.

[7] Beullens, W.: **Improved cryptanalysis of UOV and Rainbow.** Cryptology ePrint Archive, Report 2020/1343, 2020. https://eprint.iacr.org/2020/1343.pdf.

[8] Beullens, W.: **MAYO: Practical Post-Quantum Signatures from Oil-and-Vinegar Maps.** Cryptology ePrint Archive, Report 2021/1144, 2021. https://eprint.iacr.org/2021/1144.pdf.

[9] Bosma, W., Cannon, J., Playoust, C.: **The Magma algebra system. I. The user language.** Journal of Symbolic Computation 24(3-4), pp. 235–265 (1997)

[10] Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang., B.Y.: **Fast exhaustive search for polynomial systems in $\mathbb{F}_2$.** In Stefan Mangard and François-Xavier Standaert, editors, CHES 2010, volume 6225 of LNCS, pages 203–218, Santa Barbara, CA, USA, August 17–20, 2010. Springer, Heidelberg, Germany.

[11] Buss, J.F., Frandsen, G.S., Shallit, J.O.: **The computational complexity of some problems of linear algebra.** Journal of Computer and System Sciences 58(3), 572– 596 (1999).

[12] Cheng, C.M., Chou, T., Niederhagen, R., Yang, B.Y.: **Solving quadratic equations with XL on parallel architectures.** In Emmanuel Prouff and Patrick Schaumont, editors, CHES 2012, volume 7428 of LNCS, pages 356–373. Springer, Heidelberg, September 2012.

[13] Cheng, C.M., Hashimoto, Y., Miura, H., Takagi, T.: **A polynomial-time algorithm for solving a class of underdetermined multivariate quadratic equations over fields of odd characteristics**. In PQCrypto'14, LNCS 8772 (2014), pp.40–58.

[14] Courtois, N., Goubin, L., Meier, W., Tacier, J.-D.: **Solving underdefined systems of multivariate quadratic equations**. In PKC'02, LNCS 2274 (2002), pp.211–227.

[15] Courtois, N., Klimov, A., Patarin, J., Shamir, A.: **Efficient algorithms for solving overdefined systems of multivariate polynomial equations.** In Bart Preneel, editor, EUROCRYPT 2000, volume 1807 of LNCS, pages 392–407, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany.

[16] Ding, J., Chen, M.S., Kannwischer, M., Patarin, J., Petzoldt, A., Schmidt, D., Yang, B.Y.: **Rainbow. NIST Post-Quantum Cryptography Standardization Round 3 Submissions**, available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions

[17] Ding, J., Schmidt, D.: **Rainbow, a new multivariable polynomial signature scheme.** In International Conference on Applied Cryptography and Network Security, pages 164–175. Springer, 2005.

[18] Ding, J., Schmidt, D.: **Solving Degree and Degree of Regularity for Polynomial Systems over a Finite Fields.** In: Fischlin, M., Katzenbeisser, S. (eds) Number Theory and Cryptography. Lecture Notes in Computer Science, vol 8260. Springer, Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-42001-6_4.

[19] Ding, J., Yang, B.Y., Chen, C.-O., Chen, M., Cheng, C.: **New differential-algebraic attacks and reparametrization of Rainbow.** In: ACNS 2008, LNCS, vol. 5037, pp. 242–257. Springer (2008).

[20] Faugère, J.C.: **A new efficient algorithm for computing Gröbner bases (F4).** Journal of Pure and Applied Algebra, 139:61–88 (1999).

[21] Faugère, J.C.: **A new efficient algorithm for computing Gröbner bases without reduction to zero (F5).** In Proceedings of the 2002 international symposium on Symbolic and algebraic computation, pages 75–83, 2002.

[22] Furue, H., Nakamura, S., Takagi, T.: **Improving Thomae-Wolf algorithm for solving underdetermined multivariate quadratic polynomial problem**. In PQC'21, LNCS 12841 (2021), pp.65–78.

[23] Garey, M.-R., Johnson, D.-S.: **Computers and intractability: a guide to the theory of NP-completeness.** W. H. Freeman (1979).

[24] Grover, L.-K.: **A fast quantum mechanical algorithm for database search.** In STOC 1996, pp. 212–219. ACM (1996).

[25] Hashimoto, Y.: **Minor improvements of algorithm to solve under-defined systems of multivariate quadratic equations.** Available at https://eprint.iacr.org/2021/1045.pdf.

[26] Hu, Y.H., Wang, L.C., Yang, B.Y.: **"A "Medium-Field" Multivariate Public-Key Encryption Scheme."** Proc. 7th Cryptographer's Track RSA Conference, volume 3860, Lecture Notes in Computer Science, pages 132-149, 2006.

[27] Kipnis, A., Patarin, J., Goubin, L.: **Unbalanced oil and vinegar signature schemes.** In Jacques Stern, editor, EUROCRYPT'99, volume 1592 of LNCS, pages 206–222. Springer, Heidelberg, May 1999.

[28] Kipnis, A., Shamir, A.: **Cryptanalysis of the oil and vinegar signature scheme.** In Hugo Krawczyk, editor, CRYPTO'98, volume 1462 of LNCS, pages 257–266. Springer, Heidelberg, August 1998.

[29] Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlè, D., Bai, S.: **CRYSTALS-DILITHIUM.** Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[30] Matsumoto, T., Imai, H.: **Public quadratic polynomial-tuples for efficient signature verification and message-encryption.** In Advances in Cryptology — EUROCRYPT 1988, volume 330 of Lecture Notes in Computer Science, pages 419–545. Christoph G. Günther, ed., Springer, 1988.

[31] Miura, H., Hashimoto, Y., Takagi, T.: **Extended algorithm for solving underdefined multivariate quadratic equations.** In PQCryoto'13, LNCS 7932 (2013), pp.118–135.

[32] NIST: **Post-quantum cryptography CSRC.** Available at https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization

[33] NIST: **Post-Quantum Cryptography: Digital Signature Schemes.** Available at https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals

[34] NIST: **Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process.** Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf

[35] Park, C.M.: **Cryptanalysis of Matrix-based UOV.** In Finite Fields and Their Applications, Volume 50, 2018, Pages 209-221, ISSN 1071-5797, https://doi.org/10.1016/j.ffa.2017.11.012.

[36] Patarin, J.: **The oil and vinegar signature scheme.** In Dagstuhl Workshop on Cryptography September, 1997.

[37] Patarin, J.: **Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP) Two New Families of Asymmetric Algorithms.** In EUROCRYPT'96, LNCS v. 1070, pp. 33-48.

[38] Petzoldt, A.: **Selecting and reducing key sizes for multivariate cryptography.**

[39] Petzoldt, A., Thomae, E., Bulygin, S., Wolf, C.: **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.

[40] Prest, T., Fouque, P. A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: **FALCON.** Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[41] Sakumoto, K., Shirai, T., Hiwatari, H.: **On Provable Security of UOV and HFE Signature Schemes against Chosen-Message Attack.** In: Yang, BY. (eds) Post-Quantum Cryptography. PQCrypto 2011. Lecture Notes in Computer Science, vol 7071. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-25405-5_5.

[42] Shor, P. W.: **Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer.** In SIAM Journal on Computing 26(5), pp. 1484-1509 (1997).

[43] Tao, C., Diene, A., Tang, S., Ding, J.: **Simple matrix scheme for encryption.** In Gaborit, P. (ed.) PQCrypto 2013. LNCS, vol. 7932, pp.231-242. Springer, Heidelberg (2013).

[44] Tan, Y., Tang, S.: **Two Approaches to Build UOV Variants with Shorter Private Key and Faster Signature Generation.** In: Lin, D., Wang, X., Yung, M. (eds) Information Security and Cryptology. Inscrypt 2015. Lecture Notes in Computer Science(), vol 9589. Springer, Cham. https://doi.org/10.1007/978-3-319-38898-4_4.

[45] Thomae, E.: **Quo Vadis Quaternion? Cryptanalysis of Rainbow over non-commutative rings.** In SCN'12, Lect. Notes Comput. Sci. 7485, pp.361–363, 2012.

[46] Thomae, E., Wolf, C.: **Solving underdetermined systems of multivariate quadratic equations**, revisited. In PKC'12, LNCS 7293 (2012), pp.156–171.

[47] Verbel, J., Baena, J., Cabarcas, D., Perlner, R., Smith-Tone, D.: **On the complexity of "Superdetermined" minrank instances.** In: Ding, J., Steinwandt, R. (eds.) PQCrypto 2019. LNCS, vol. 11505, pp. 167–186. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_10

[48] Wang, L.C., Chang, F.H.: **Tractable Rational Map Cryptosystem** Available at http://eprint.iacr.org/2004/046.pdf.

[49] Wang, L.C., Hu, Y.H., Lai, F., Chou, C.Y., Yang, B.Y.: **Tractable rational map signature.** In PKC, Serge Vaudenay, ed., Public Key Cryptography — PKC 2005, (2005), pages 244–257. ISBN 3-540-24454-9.

[50] Wang, L.C., Tseng, P.E., Kuan, Y.L., Chou, C.Y.: **NOVA, a Noncommutative-ring Based Unbalanced Oil and Vinegar Signature Scheme with Key-randomness Alignment**, 2022. Available at https://eprint.iacr.org/2022/665.

[51] Wang, L.C., Tseng, P.E., Kuan, Y.L., Chou, C.Y.: **A Simple Noncommutative UOV Scheme**, 2022. Available at https://eprint.iacr.org/2022/1742.

[52] Wang, L.C., Wei, T.J., Shih, J.M., Hu, Y.H., Hsieh, C.C.:**An algorithm for solving over-determined multivariate quadratic systems over finite fields.** doi: 10.3934/amc.2022001

[53] Wiedemann, D.: **Solving sparse linear equations over finite fields.** IEEE Trans. Inf. Theory IT-32, pp. 54-62, 1986.

[54] Wiggers, T.: **Making protocols post-quantum.** In the Cloudflare blog. Available at https://blog.cloudflare.com/making-protocols-post-quantum/

[55] Westerbaan, B.: **Sizing Up Post-Quantum Signatures.** In the Cloudflare blog. Available at https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

[56] Yasuda, T., Sakurai, K., Takagi, T.: **Reducing the Key Size of Rainbow Using Non-Commutative Rings.** In CT-RSA, volume 7178 of Lecture Notes in Computer Science, pages 68-83. Springer, 2012.