SNOVA

Proposal for NISTPQC: Additional Digital Signature Schemes

Version 2 + Odd q

Lih-Chung Wang, Chun-Yen Chou, Jintai Ding, Yen-Liang Kuan,
Jan Adriaan Leegwater, Ming-Siou Li, Bo-Shu Tseng,
Po-En Tseng, Chia-Chun Wang

September 2, 2025

pqclaborg@gmail.com
https://snova.pqclab.org

Changelog

Version 2 + Odd q: Add specification for fields of odd prime order with a symmetric public matrix.

Version 2.0: In the following, we describe the changes between the Round 1 submission of SNOVA dated May 25, 2023, and the Round 2 adjustments, along with brief explanations of the motivations behind the changes.

Additional Submitters

We add one new member to our team: Jan Adriaan Leegwater.

Changes to the SNOVA Public Map

Round 2 SNOVA public map. These changes are motivated by the cryptanalysis that has been performed during Round 1. For $i \in \{1, \ldots, m\}$, the public map of SNOVA will be:

$$\widetilde{P}_i(\mathbf{U}) = \widetilde{F}_i(T(\mathbf{U})) = \sum_{\alpha=0}^{l^2+l-1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{i,\alpha} \cdot U_j^t (Q_{i,\alpha,1} P_{i',jk} Q_{i,\alpha,2}) U_k \cdot B_{i,\alpha}$$

where $i' = i + \alpha \mod m$.

- Mixing P_i of the public map
- Increasing the number of terms in the summation over α. More precisely, we changed the index of the summation from $\alpha = 1, \ldots, l^2$ to $\alpha = 0, \ldots, l^2 + l - 1$.
- Varying ABQ matrices (which refers to $A_{i,\alpha}, B_{i,\alpha}, Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ matrices in the public map of SNOVA) also with index i.
- Using fixed ABQ matrices when $l \leq 3$.

In the Round 1 evaluation, SNOVA met NIST's security requirements against key recovery attacks (with slight adjustment on $l = 2$ parameters). The primary threats to SNOVA's security were forgery attacks proposed by Beullens and Cabarcas *et al.*, as discussed in Section 5.2.3 and Section 5.2.4. However, both attacks are highly sensitive to the rank of the $\mathbf{E_R}$ matrix. When the $\mathbf{E_R}$ matrix does not experience significant rank reduction, the efficiency of these attacks diminishes considerably. This is precisely the case for Round 2 SNOVA. Consequently, SNOVA's security is no longer compromised by these attacks.

Changed and Added Parameters

- **Update vinegar parameters for $l = 2$ as a response to cryptanalysis.**

Table 1: SNOVA $l = 2$ parameter sets

Security Level	Round 1	Round 2
I	$(28, 17, 16, 2)$	$(37, 17, 16, 2)$
III	$(43, 25, 16, 2)$	$(56, 25, 16, 2)$
V	$(61, 33, 16, 2)$	$(75, 33, 16, 2)$

- **Add $l = 5$ parameter sets as these offer attractive public key sizes.**

Table 2: SNOVA $l = 5$ parameter sets

Security Level	$l = 5$ parameter sets
III	$(24, 5, 16, 5)$
V	$(29, 6, 16, 5)$

Other Changes

- **Option for the public key expansion.** We add the option to use SHAKE for public key expansion.

- **AVX2 implementation.** We have implemented an AVX2 version of SNOVA, which significantly improves system execution efficiency compared to the Round 1 SNOVA implementation. For more information, we refer to Table 10.

- **Updated security analysis.** We have updated our security analysis. During Round 1 evaluation, a large number of papers with cryptanalysis of SNOVA were published [34, 37, 11, 16, 41, 2]. We explored and adjusted our analysis to address the known attack methods. In the updated security analysis, we provide the complexity corresponding to each attack. The Round 2 security analysis also focuses on the forgery attacks against SNOVA. In the observed forgery attacks [8, 16], a crucial point is the rank of $\mathbf{E_R}$ matrix in the attack. With the adjustments made in Round 2, we found that neither of these forgery attacks poses a threat to the security of SNOVA.

- **Editorial changes to the specification.** We corrected typos and made the notation in the document more consistent.

Contents

1 Algorithm Specification (2.B.1)

In this supporting document, we present a detailed specification of multivariate signature scheme SNOVA: **S**imple **N**oncommutative unbalanced **O**il and **V**inegar scheme with randomness **A**lignment. In the following sections, Introduction and Preliminaries, we adapt and slightly modify paragraphs from Wang *et al.* [64, 65].

1.1 Introduction

SNOVA is a variant of the Unbalanced Oil and Vinegar (UOV) signature scheme, designed to operate over noncommutative rings for enhanced efficiency and reduced public key size.

Unbalanced Oil and Vinegar. The Unbalanced Oil and Vinegar (UOV) signature scheme [35] is a slight modification of the Oil and Vinegar (OV) [48] signature scheme, proposed by Patarin in 1997. The UOV signature scheme has been studied and analyzed for a long time. To this day, it is still believed to be a secure scheme. However, as a multivariate signature scheme, it still suffers from the problem of having excessively large public keys. In the literature, fundamental public key compression methods have been proposed. A. Petzoldt [50, 51] and Rainbow [21] of the third-round of NIST proposal showed that 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 (Pseudorandom Number Generator) which we called "randomness alignment" technique here. This reduces the public key size of UOV to the order $O(m^3 \cdot \log q)$.

For the modern parameters of UOV which aiming at NIST security level I [44], the public key sizes are about 40KB to 60KB. However, these public key sizes of the UOV scheme are still too large. To alleviate this problem, new possibilities have come to light. By generalizing the UOV scheme to noncommutative rings, we can further reduce the size of the public key.

SNOVA signature scheme. SNOVA is a variant of UOV with smaller public key sizes. In SNOVA, we see several advantages:

- By building on noncommutative rings, we can reduce the size of the public key while still maintaining the advantage of short signatures.
- The randomness alignment key-compression technique of Petzoldt [50] can be successfully adapted to SNOVA without being affected by noncommutativity.
- There is an intuitive connection between SNOVA and UOV. In the case that $l = 1$ of the underlying matrix ring, SNOVA reduces to a UOV scheme.

We propose parameter settings aiming for NIST security levels I, III, and V. For security level I, our $l = 4$ parameter set results in a public key size of 1016 bytes and a signature size of 248 bytes. With these performances, we believe that the SNOVA scheme has strong competitiveness compared to other post-quantum signature schemes.

1.2 Preliminaries

1.2.1 Notations and Conventions

The following Tables 3, 4 are tables that list some symbols fixed with specific meaning and some conventions on notations, respectively.

Table 3: The table of symbols fixed with specific meaning 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 its characteristic polynomial irreducible over $\mathbb{F}_q$
$n = v + o$	number of variables
$m = o$	number of equations
$F = [F_1, \cdots, F_m]$	central map of the ring UOV scheme
$\widetilde{F} = \left[\widetilde{F}_1, \cdots, \widetilde{F}_m\right]$	central map of the SNOVA scheme
T	invertible linear map in signature scheme
$[T]$	matrix corresponding to T
$P = [P_1, \cdots, P_m]$	public map of the ring UOV scheme
$\widetilde{P} = \left[\widetilde{P}_1, \cdots, \widetilde{P}_m\right]$	public map of the SNOVA scheme
$\mathcal{O}$	oil space
$MQ(N, M, q)$	complexity of an MQ (Multivariate Quadratic) system of M equations in N variables over $\mathbb{F}_q$
$\mathbf{A}^{\otimes n}$	the block diagonal matrix with n copies of A on the block diagonal

Table 4: The table of conventions on notations 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	$\mathbf{X}$ and $\mathbf{x}$
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 $P_{i,jk}$

1.2.2 Basic Notions

MQ problem. Let $\mathbb{F}_q$ be a finite field of order q. Given M quadratic polynomials $P(\mathbf{x}) = [P_1(\mathbf{x}), \cdots, P_M(\mathbf{x})]$ in N variables $\mathbf{x} = (x_1, \cdots, x_N)^t$ and a vector $\mathbf{y} \in \mathbb{F}_q^M$, the MQ (Multivariate Quadratic) problem is to find a vector $\mathbf{u} \in \mathbb{F}_q^N$ such that $P(\mathbf{u}) = [P_1(\mathbf{u}), \cdots, P_M(\mathbf{u})] = \mathbf{y}$. This problem is known to be NP-hard when $N \sim M$ [30]. 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 $N \geq M(M+1)$ [8].

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 [25], F_5 [26] and XL variants [20, 66].

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

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

where for each $i \in \{1, \ldots, M\}$ the polar form of $P_i(\mathbf{x})$ is defined by

$$P'_i(\mathbf{x}, \mathbf{y}) = P_i(\mathbf{x} + \mathbf{y}) - P_i(\mathbf{x}) - P_i(\mathbf{y}).$$

Note that each P'_i is symmetric and bilinear. If we write the quadratic map into the form of $P_i(\mathbf{x}) = \mathbf{x}^t [P_i] \mathbf{x}$ where $[P_i]$ is the matrix representation of P_i then the matrix representation of P_i' is

$$[P_i'] = [P_i] + [P_i]^t .$$

1.2.3 NIST Security Level.

In [43], 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 a 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 5: 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 [35] signature scheme is a slight modification of the Oil and Vinegar (OV) [48] 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 KS attack [36] 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 the 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 $P_i = \mathbf{u}^t [P_i] \mathbf{u}$ where $\mathbf{u} = (u_1, \cdots, u_n)^t$ 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 the UOV scheme indicates that F vanishes on the linear space $\mathcal{O} = \{\mathbf{x} = (x_1, \ldots, x_n)^t \in \mathbb{F}_q^n : x_1 = \cdots = x_v = 0\}$ called the oil space of central map F, and hence the oil space of public key P will be the space $T^{-1}(\mathcal{O})$.

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

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 [25, 26, 20]. 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 [35] 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 $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$.

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. The multiplication with other random matrices and then summing together happens to scramble the entries in $\mathbb{F}_q$ and hence make the sparsity of the matrix multiplication 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 (which may be called key-randomness alignment) [50] 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.

In the first round of evaluation, several attacks were proposed [34, 37, 11, 16, 41, 2]. We are very grateful for these analyses, and building on them, we made some minor adjustments. After these adjustments, we believe that SNOVA remains highly competitive while meeting NIST security requirements. Additionally, we have introduced some new $l = 5$ parameter sets, which not only show promising performance but also bring new possibilities. On the other hand, by proposing a potential alternative, we would like to demonstrate the flexibility of the SNOVA.

1.5 Ring UOV

In order to enhance the comprehension of SNOVA, we now introduce an intermediary phase called ring UOV, which generalizes UOV to any noncommutative ring $\mathcal{R}$. Other schemes using noncommutative rings with different techniques have been proposed [27, 70]. Similar to UOV, let $n = v + o$ and $m = o$. Due to the noncommutativity of $\mathcal{R}$ we need to explicitly denote the following index set which will be used below by

$$\Omega = \{(j,k) : 1 \le j,k \le n\} \setminus \{(j,k) : v+1 \le j,k \le n\}.$$

The basic structure of ring UOV. The central map of ring UOV is the map $F = [F_1, \cdots, F_m] : \mathcal{R}^n \to \mathcal{R}^m$ with each F_i defined by

$$F_i(X_1, \ldots, X_n) = \sum_{(j,k)\in\Omega} \phi(X_j) F_{i,jk} X_k$$

where the coefficients $F_{i,jk}$ are randomly chosen from $\mathcal{R}$. The map ϕ is a ring map with "factor order reversed" property, i.e., $\phi\left(\sum_j C_j X_j\right) = \sum_j \phi(X_j)\phi(C_j)$ where $C_j \in \mathcal{R}$. The (ring) variables $X_1, \ldots, X_v$ are called the vinegar variables and $X_{v+1}, \cdots, X_n$ are called the oil variables.

A concrete example of ring UOV. For the purpose of explaining SNOVA, we now fix the noncommutative ring to be $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$ and the ring map ϕ to be the matrix transpose. Then, we have a (v, o, q, l)-ring UOV scheme.

Due to these specification, the i-th component, for $i \in \{1, 2, \ldots, m\}$, of the central map $F = [F_1, \cdots, F_m] : \mathcal{R}^n \to \mathcal{R}^m$ becomes

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

Note that we can write F_i into quadratic form over $\mathcal{R}$. That is,

$$F_i(\mathbf{X}) = \mathbf{X}^t \left[F_i\right] \mathbf{X}$$

where $\mathbf{X} = (X_1, \ldots, X_n)^t$ and the matrix representation $[F_i]$ over $\mathcal{R}$ corresponding to F_i is of the form

$$[F_i] = \left[F_{i,jk}\right] = \begin{bmatrix} F_i^{11} & F_i^{12} \\ F_i^{21} & 0 \end{bmatrix},$$

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.

The public map $P = [P_1, \cdots, P_m]$ is the composition of central map F and an invertible ring linear map $T : \mathcal{R}^n \to \mathcal{R}^n$, i.e.,

$$P(\mathbf{U}) = (F \circ T)(\mathbf{U})$$

where $P_i(\mathbf{U}) = (F_i \circ T)(\mathbf{U})$ for each $i \in \{1, 2, \ldots, m\}$.

The map T is defined by its matrix representation

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

where T^{12} is a $v \times o$ random matrix over $\mathcal{R}$ and I^{11}, I^{22} are identity matrices over $\mathcal{R}$ of size $v \times v$ and $o \times o$, respectively.

Public key and private key. For each $i \in \{1, \ldots, m\}$, we have

$$P_i(\mathbf{U}) = (F_i \circ T)(\mathbf{U}) = \mathbf{U}^t \Big([T]^t [F_i] [T] \Big) \mathbf{U}.$$

Therefore, the public keys are $[P_1], \ldots, [P_m]$ where

$$[P_i] = \left[P_{i,jk}\right] = [T]^t [F_i] [T]$$

for $i \in \{1, \cdots, m\}$. The private key is (F, T), i.e., the matrix $[T]$ and the matrices $[F_i]$.

1.6 SNOVA Signature Scheme

In this section, we introduce SNOVA signature scheme whose central map is a modified ring UOV map. In order to eliminate the sparsity of ring UOV map (when we regard it as a UOV map over field), some specific matrices will be introduced into the ring UOV map.

1.6.1 Description

Let v, o be positive integers with $v > o$ and $\mathbb{F}_q$ be a field of order q. Next, we will introduce the subring of the matrix ring $\mathcal{R}$, $\mathbb{F}_q[S]$. Last, we will define a (v, o, q, l) SNOVA scheme.

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

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

As the characteristic polynomial of S is irreducible, $\mathbb{F}_q[S]$ is a field. Thus, the elements in $\mathbb{F}_q[S]$ are symmetric and they all commute.

Let

$$\Omega = \{(j,k) : 1 \leq j,k \leq n\} \setminus \{(j,k) : v+1 \leq j,k \leq n\}.$$

This index set Ω is defined by the Oil-Vinegar structure.

Central map. For $i \in \{1, \ldots, m\}$, we define

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

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

We use random matrices and $[F_1], \ldots, [F_m]$ to construct the central map.

First, we randomly choose invertible elements $A_{i,\alpha}$ and $B_{i,\alpha}$ from $\mathcal{R}$, and invertible elements $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ from $\mathbb{F}_q[S]$.

The central map of SNOVA scheme is $\widetilde{F} = \left[\widetilde{F}_1, \cdots, \widetilde{F}_m\right] : \mathcal{R}^n \to \mathcal{R}^m$, for $i \in \{1, \cdots, m\}$, $\widetilde{F}_i$ is defined to be

$$\widetilde{F}_i(X_1, \ldots, X_n) = \sum_{\alpha=0}^{l^2+l-1} A_{i,\alpha} \cdot \left(\sum_{(j,k)\in\Omega} X_j^t \left(Q_{i,\alpha,1} F_{i',jk} Q_{i,\alpha,2}\right) X_k \right) \cdot B_{i,\alpha}$$

where $i' = (i + \alpha) \mod m$ and $F_{i',jk}$ is the (j,k)-th entry of $[F_{i'}]$. Changes to the Round 1 central map are: the sum over α has been extended from l^2 to l^2+l elements, the ABQ matrices now depend not only on α but also on i, and $F_{i'}$ depends on both i and α.

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

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

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 $\mathbb{F}_q[S]$. In particular, T_{ij} commutes with $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$. The matrices I^{11} and I^{22} are identity matrices over $\mathcal{R}$. Therefore, $[T]$ is invertible and hence T. Note that

$$\left[T^{-1}\right] = \begin{bmatrix} I^{11} & T^{12} \\ 0 & I^{22} \end{bmatrix},$$

Public map. We construct the public key $[P_1], \ldots, [P_m]$ via the congruence relation

$$[P_1] = \left[P_{1,j,k}\right] = [T]^t [F_1] [T], \ldots, [P_m] = \left[P_{m,j,k}\right] = [T]^t [F_m] [T].$$

The public map of SNOVA is $\widetilde{P} = \widetilde{F} \circ T$. For $i \in \{1, 2, \ldots, m\}$, $\widetilde{P}_i = \widetilde{F}_i \circ T$. Then, by the relation $\mathbf{X} = [T] \cdot \mathbf{U}$ where $\mathbf{U} = (U_1, \cdots, U_n) \in \mathcal{R}^n$ and the commutativity of $\mathbb{F}_q[S]$, we have that

$$\widetilde{P}_i(\mathbf{U}) = \widetilde{F}_i(T(\mathbf{U})) = \sum_{\alpha=0}^{l^2+l-1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{i,\alpha} \cdot U_j^t (Q_{i,\alpha,1} P_{i',jk} Q_{i,\alpha,2}) U_k \cdot B_{i,\alpha}$$

where $i' = (i + \alpha) \mod m$ and $P_{i',jk}$ is the (j,k)-th entry of matrix $[P_{i'}]$. By introducing the matrices $A_{i,\alpha}, B_{i,\alpha}, Q_{i,\alpha,1}, Q_{i,\alpha,2}$, the public map $\widetilde{P}$ is not a sparse UOV map when we regard it as over $\mathbb{F}_q$.

Public key. The public key are the matrices $[P_i]$ and the matrices $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $\alpha = 0, 1, \ldots, l^2+l-1$, or simply the seed $\mathbf{s_{public}}$ which generates them. By utilizing matrices $[P_i]$ and the seed $\mathbf{s_{public}}$, the verifier is capable to obtain the public map $\widetilde{P}$ and subsequently verify the received signature.

Private key. The private key of SNOVA is (F, T), i.e., the matrix $[T]$ and the matrices $[F_i]$ for $i = 1, 2, \ldots, m$. Note that we can use the private seed $\mathbf{s_{private}}$ to generate T.

Signature. Let D be the document to be signed and $Hash(D) = \mathbf{Y} = (Y_1, \cdots, Y_m) \in \mathcal{R}^m$ be its hash value. We compute the signature $\mathbf{U}$ 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 by regarding SNOVA as a UOV over $\mathbb{F}_q$. Secondly, the signature is $\mathbf{U} = T^{-1}(\mathbf{X}) \in \mathcal{R}^n$.

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

1.6.2 Key Generation Process of SNOVA

We give the standard key generation process of SNOVA and the key generation process with key-randomness alignment technique.

Standard key generation process. For $i \in \{1, \dots, m\}$, the matrix $[P_i]$ is obtained by the relation

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

As $F_i^{22} = 0$, we have the following relations

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

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 $i \in \{1, \dots, m\}$ 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 $i \in \{1, \dots, m\}$ from public seed $\mathbf{s_{public}}$. Generate $[T]$ from private seed $\mathbf{s_{private}}$. We also generate the matrices $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $\alpha = 0, 1, \dots, l^2 + l - 1$ from $\mathbf{s_{public}}$ or, if $l \leq 3$, we generate these matrices from a fixed seed.

Second Step: Compute the matrix $F_i^{11}, F_i^{12}, F_i^{21}, P_i^{22}$ for $i = 1, \cdots, m$ as below. Inverting the relation between F and P above, the following equations hold

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

In other words, we have

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

1.6.3 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 [45, 53].

Signature. Let D be the document to be sign, we randomly choose **salt** and then generate a signature for the hash value $\mathbf{Y} = Hash(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$ 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 $\sigma = (\mathbf{U}||\mathbf{salt})$ where $\mathbf{U}$ is the signature of $\mathbf{Y}$ generated by the SNOVA signer. While there is no immediate security risk if salts are used more than once, each signature generated should use a different random value of **salt**. 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 [44].

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

1.7 Implementation Details

In this section, we describe the details about implementations.

For all parameter settings at $q = 16$ 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 5, 7 and 10 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 intends to sign. That is, the **Algorithm**s 5, 7 and 10 are only used in key generation process.

For odd q we only use the SNOVA-ssk variant.

Symmetric public matrix. The public matrix is taken to be symmetric when the characteristic of the field $\mathbb{F}_q$ is odd. This allows for a significant the reduction of the public key size.

Symmetric primitives. In SNOVA scheme, several hash functions and PRNG are needed. An implementation may choose to introduce a function to convert a seeded secret key into an expanded secret key if multiple signatures are to be created. We consider this an implementation choice and not a part of the specification of SNOVA. 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 or SHAKE128.
- 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.

A variant that uses SHAKE128 (Secure Hash Algorithm KECCAK) for the public key expansion has been added in the Round 2 submission as an alternative to AES128-CTR. This variant can be both vectorized and indexed. We denote the XOF (eXtendable Output Function) of this variant as SNOVA_SHAKE. The algorithm of SNOVA_SHAKE can be described by: extract the bytes from SHAKE128 in 168 bytes blocks where a block index is appended to the seed. The block size 168 follows from the rate of SHAKE128. SNOVA_SHAKE is specified by: Let `SHAKE128(seed, n)` denote the n-th byte of the `SHAKE128` XOF when instantiated with `seed` as input, and similarly `SNOVA_SHAKE(seed, n)`. Then for all required bytes:

```
SNOVA_SHAKE(seed, n) = SHAKE128(seed || floor(n / 168), (n mod 168))
```

where `floor(n / 168)` is the 8 bytes little-endian representation of `n / 168` rounded to below, `||` represents concatenation of bytes and `mod` is the integer modulus operation.

1.8 Constants and Tables

The finite field $\mathbb{F}_q$. For prime q this field is defined by usual arithmetic modulo q.

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 the 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

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

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

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

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:

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

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

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

$$S = \begin{bmatrix} 8 & 7 & 6 & 5 & 4 \\ 7 & 6 & 5 & 4 & 3 \\ 6 & 5 & 4 & 3 & 2 \\ 5 & 4 & 3 & 2 & 1 \\ 4 & 3 & 2 & 1 & 9 \end{bmatrix} \quad \text{if } l = 5.$$

One can check that the characteristic polynomials of these matrices S are irreducible over $\mathbb{F}_{16}$.

The matrix $\boldsymbol{S_q}$. For prime q we use the following sets of matrices:

$$\begin{aligned} S(i,j) &= (q_a + i + j) \textbf{ and } q_b \qquad && (i \neq l-1) \vee (j \neq l-1) \\ S(i,j) &= q_c && (i = l-1) \wedge (j = l-1) \end{aligned}$$

where **and** is a bitwise and. For the parameters q_a, q_b, q_c we use the values in Table 6. It can be verified that for $l = 2, ..., 5$ all S matrices are irreducible.

Table 6: Parameters of the S matrix depending on the field order q.

q	q_a	q_b	q_c
7	4	6	1 (if $l \neq 3$) 5 (if $l = 3$)
11	0	3	6
13	2	11	3
17	1	11	10
19	1	3	15
23	1	11	22
29	3	12	11
31	2	5	8

This results in for example:

$$S = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \\ 0 & 1 & 2 & 15 \end{bmatrix} \quad \text{if } q = 19 \text{ and } l = 4.$$

$$S = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 3 & 0 & 1 & 2 \\ 0 & 1 & 2 & 22 \end{bmatrix} \quad \text{if } q = 23 \text{ and } l = 4.$$

1.9 Algorithms

1.9.1 Shared Algorithms

Here we decribe algorithms that are used by all functions.

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

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

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

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

Generate invertible matrices. Let $l = 2, 3, 4, 5$ and $M \in \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$ any $l \times l$ matrix over $\mathbb{F}_q$. Since the polynomial $\det(M + xS)$ in the variable x has at most l roots, there exists an element a of $\mathbb{F}_q$ 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 ← 1 to q − 1 do
        if det(M + aS) ≠ 0 then
            M ← M + aS
            break
        end
    end
end
return M
```

Conversion between field elements and bytes. To convert N_q elements of $\mathbb{F}_q$ with coefficients $c_{q,i}$ into N_b bytes $c_{b,i}$ we calculate the sum

$$s_q = \sum_{i=0}^{N_q-1} c_{q,i} q^i,$$

and store the result as bytes, least significant byte first. This relation can easily be inverted. The values of N_q and N_b are such that any sequence of $\mathbb{F}_q$ can be converted to bytes. The reverse does not hold. If the sum of bytes is too large, so

$$s_b = \sum_{i=0}^{N_b-1} c_{b,i} q^i \geq q^{N_q},$$

the input provided (public key or signature) is considered illegal, and the calling algorithm will exit with an error.

SNOVA uses the following values for N_q and N_b:

Table 7: Parameters of conversion $\mathbb{F}_q \Leftrightarrow$ bytes. The last two columns present the minimal number of bits required to encode an $\mathbb{F}_q$ element, and the actual number of bits used.

q	N_q	N_b	$\log_2(q)$	$8N_b/N_q$
7	17	6	2.807	2.823
11	16	7	3.459	3.5
13	15	7	3.7	3.733
16	2	1	4	4
17	15	8	4.087	4.267
19	15	8	4.248	4.267
23	7	4	4.524	4.571
29	13	8	4.858	4.923
31	8	5	4.954	5

Note that for $q = 16$ this is the nibble ordering used in Rounds 1 and 2. The specified conversion allows for an efficient packing, reducing the public key and signature sizes.

Algorithm 3: Convert elements of $\mathbb{F}_q$ to bytes

input : n elements $a_0, \ldots, a_{n-1}$ of $\mathbb{F}_q$
output: a byte string

$x \leftarrow 0^0$
for $j \leftarrow 0$ **to** $(n-1)/N_q$ **do**
    $s_q \leftarrow \sum_{i=0}^{N_q-1} q^i a_{i+jN_q}$
    $y \leftarrow s_q$ (as radix 256 number, N_b bytes, least significant first)
    $x \leftarrow x||y$
end
return x

Algorithm 4: Convert bytes to elements of $\mathbb{F}_q$

input : n bytes $b_0, \ldots, b_{n-1}$
output: a string of elements of $\mathbb{F}_q$

$x \leftarrow 0^0$
for $j \leftarrow 0$ **to** $(n-1)/N_b$ **do**
    $s_{256} \leftarrow \sum_{i=0}^{N_b-1} (256)^i b_{i+jN_b}$
    $y \leftarrow s_{256}$ (as radix q number, N_q values, least significant first)
    $x \leftarrow x||y$
end
return x

1.9.2 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. 1.8. Generate $[T]$ from private seed $\mathbf{s_{private}}$. Generate P_i^{11}, P_i^{12} and P_i^{21} for $0 \leq i < m$ from public seed $\mathbf{s_{public}}$. We also generate the matrices $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$, and P_i^{11}, P_i^{12} and P_i^{21} for $0 \leq i < m$ from $\mathbf{s_{public}}$.

Algorithm 5: 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})$\leftarrow Hash_{\text{SHAKE256}}(\mathbf{s_{private}})$
▷ Use bytes as nibbles if $q = 16$.
▷ Use value $\mod q$.
▷ Rejection sampling: skip values close to 256.
▷ $Hash_{\text{SHAKE256}}$ is instantiated as SHAKE256 throughout
Generate entries of T^{12} using **Algorithm** 1
return $[T^{12}]$

Algorithm 6: `SNOVA_SHAKE` public key expansion

input : Seed M
The requested number of bytes N
output: Pseudo-random bytes

$x \leftarrow 0^0$ ▷ x is set to the empty string
$b \leftarrow 0$
$n \leftarrow 0$
while $n < N$ **do**
  $d \leftarrow \texttt{min}(168, N - n)$
  $x \leftarrow x \parallel \texttt{SHAKE128}(M \parallel b_{64}, d)$
  $n \leftarrow n + d$
  $b \leftarrow b + 1$
end
return x

Algorithm 7: Generate the random part of public key for $q = 16$

input : SNOVA parameters (v, o, l)
public seed $\mathbf{s_{public}}$

output: the matrices ($A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$)
the matrices (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)

if $l \leq 3$ **then**
  (entries of $(P_0^{11}||\cdots||P_{m-1}^{11})||(P_0^{12}||\cdots||P_{m-1}^{12})||(P_0^{21}||\cdots||P_{m-1}^{21})$) $\leftarrow Hash_{\text{AES128}}(\mathbf{s_{public}})$
  $\mathbf{s_{fixed}} \leftarrow$ "SNOVA_ABQ"
  (entries of $(A_0||\cdots||A_{l^2+l-1})||(B_0||\cdots||B_{l^2+l-1}))||$ (coefficients of S-polynomials for entries in the concatenation $(Q_{0,1}||\cdots||Q_{(l^2+l-1),1})||(Q_{0,2}||\cdots||Q_{(l^2+l-1),2})$) $\leftarrow$ SHAKE256($\mathbf{s_{fixed}}$)
else
  (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 $(A_0||\cdots||A_{l^2+l-1})||(B_0||\cdots||B_{l^2+l-1}))||$ (coefficients of S-polynomials for entries in the concatenation $(Q_{0,1}||\cdots||Q_{(l^2+l-1),1})||(Q_{0,2}||\cdots||Q_{(l^2+l-1),2})$) $\leftarrow Hash_{\text{AES128}}(\mathbf{s_{public}})$
end
  ▷ $Hash_{\text{AES128}}$ is instantiated as AES128 throughout
  ▷ $Hash_{\text{SNOVA\_SHAKE}}$ (Algorithm 6) is an alternative to $Hash_{\text{AES128}}$
for $\alpha \leftarrow 0$ **to** $l^2 + l - 1$ **do**
  let $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ be invertible using **Algorithm** 2
end
return ($A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$) and (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)

Algorithm 8: Generate the random part of public key for odd q

input : SNOVA parameters (v, o, l)
public seed $\mathbf{s_{public}}$
output: the matrices ($A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$)
the matrices (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)

(upper triangular 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 $(A_0||\cdots||A_{l^2+l-1})||(B_0||\cdots||B_{l^2+l-1}))||$
1 (coefficients of S-polynomials for entries in the concatenation
$(Q_{0,1}||\cdots||Q_{(l^2+l-1),1})||(Q_{0,2}||\cdots||Q_{(l^2+l-1),2}))$
$\leftarrow Hash_{\text{SNOVA\_SHAKE}}(\mathbf{s_{public}})$
2 **for** $\alpha \leftarrow 0$ **to** $l^2 + l - 1$ **do**
3 let $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ be invertible using **Algorithm** 2
4 **end**
5 Make P symmetric
6 **return** ($A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$) and (P_i^{11}, P_i^{12}, P_i^{21} for $0 \leq i < m$)

Second Step: Compute the matrix P_i^{22} for $0 \leq i < m$. As noted above, we have

$$F_i^{12} = P_i^{12} + P_i^{11} T^{12}$$
$$P_i^{22} = -(T^{12})^t F_i^{12} - P_i^{21} T^{12}$$

The public key expansion algorithm is given in **Algorithm** 7. Beullens has shown in [11] that some public keys of the Round 1 version of SNOVA are weak. We found that this applies even when including the Round 2 tweaks if $l \leq 3$ albeit in a much weaker way. Refer to Section 5.2.3 for more information. At $l = 2$, we estimate that the probability of arriving at a "weak" key with ABQ matrices derived from the public key is less than 2^{-48}, which we believe is still unacceptable. As an additional measure, we fix the seed used to generate the ABQ matrices to some arbitrary but "well-chosen" value if $l \leq 3$. We use the ASCII string "SNOVA_ABQ" and SHAKE256 for the generation of the ABQ matrices if $l \leq 3$. We have verified that this fixed seed for $l = 2, 3$ results in a set of ABQ matrices that has a MinRank of $l^2 o - l + 1$ (see Section 5.2.3).

Algorithm 9: 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** 5
$m \leftarrow o$
Generate $(P_i^{11}, P_i^{12}, P_i^{21}$ for $0 \leq i < m)$ using **Algorithm** 7
for $i \leftarrow 0$ **to** $m - 1$ **do**
  $F_i^{12} \leftarrow P_i^{12} + P_i^{11}T^{12}$
  $P_i^{22} \leftarrow -\left(T^{12}\right)^t F_i^{12} - P_i^{21}T^{12}$
end
return $(\mathbf{s_{public}}, P_i^{22}$ for $0 \leq i < m)$ as bytes
  ▷ If $q \neq 16$: Return the upper triangular part of P_{22}.

1.9.3 Algorithms for Signature Generation

Algorithm 10: Expand 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** 5
$m \leftarrow o$
Generate $(P_i^{11}, P_i^{12}, P_i^{21}$ for $0 \leq i < m)$ using **Algorithm** 7
for $i \leftarrow 0$ **to** $m - 1$ **do**
  $F_i^{11} \leftarrow P_i^{11}$
  $F_i^{12} \leftarrow P_i^{12} + P_i^{11}T^{12}$
  $F_i^{21} \leftarrow P_i^{21} + (T^{12})^t P_i^{11}$
end
return $(T^{12}, F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \leq i < m)$

Let D be the document to be signed, we randomly choose a **salt** and then generate a signature for the hash value $\mathbf{Y} = Hash(\mathbf{s_{public}}||Hash(D)||\mathbf{salt})$ 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 signs $\mathbf{num_{sig}}$ as follows

Algorithm 11: Assign values to vinegar variables

input : SNOVA parameters (v, o, l)
digest of the document **digest** $= Hash(D)$
the number of sign $\mathbf{num_{sig}}$
salt

output: vinegar values $(X_0, \ldots, X_{v-1})$

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

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

$$\widetilde{F}_{i,VV} = \sum_{\alpha=0}^{l^2+l-1} A_{i,\alpha} \cdot \left(\sum_{j=0}^{v-1} \sum_{k=0}^{v-1} X_j^t \left(Q_{i,\alpha,1} F_{i',jk} Q_{i,\alpha,2} \right) X_k \right) \cdot B_{i,\alpha}$$

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

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

and we also write

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

We can now compute the vinegar part values $\widetilde{F}_{i,VV}$ of the central map $\widetilde{F}_i$ by the following algorithm.

Algorithm 12: 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 ($A_{i,\alpha}, B_{i,\alpha}, Q_{i,\alpha,1}, Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l$)
vinegar values $(X_0, \ldots, X_{v-1})$
output: the vinegar part ($\widetilde{F}_{i,VV}$ for $0 \leq i < m$)

$m \leftarrow o$
for $i \leftarrow 0$ **to** $m-1$ **do**
  $\widetilde{F}_{i,VV} \leftarrow 0$
end
for $\alpha \leftarrow 0$ **to** $l^2 + l - 1$ **do**
  for $i \leftarrow 0$ **to** $m-1$ **do**
    $i' \leftarrow (i + \alpha) \mod m$
    for $j \leftarrow 0$ **to** $v-1$ **do**
      $\mathbf{Left}_{i,\alpha}[j] \leftarrow A_{i,\alpha} * X_j^t * Q_{i,\alpha,1}$ ▷ the left term of $F_i^{11}[j][k]$
      $\mathbf{Right}_{i,\alpha}[j] \leftarrow Q_{i,\alpha,2} * X_j * B_{i,\alpha}$ ▷ the right term of $F_i^{11}[j][k]$
    end
    for $j \leftarrow 0$ **to** $v-1$ **do**
      for $k \leftarrow 0$ **to** $v-1$ **do**
        $\widetilde{F}_{i,VV} \leftarrow \widetilde{F}_{i,VV} + \mathbf{Left}_{i,\alpha}[j] * F_{i'}^{11}[j][k] * \mathbf{Right}_{i,\alpha}[k]$
      end
    end
  end
end
return ($\widetilde{F}_{i,VV}$ for $0 \leq i < m$)

The resulting system can be seen as a linear system of the oil variables over the finite field $\mathbb{F}_q$. 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 $\mathbf{L} = (\mathbf{l}_{t_i t_j})_{0 \leq t_i, t_j < l}$ and $\mathbf{R} = (\mathbf{r}_{t_i t_j})_{0 \leq t_i, t_j < l} \in \mathcal{R}$. Let $X = (x_{t_i t_j})_{0 \leq t_i, t_j < l}$ be a $l \times l$ matrix with variables $x_{t_i t_j}$. Write

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

where $M = \left(m_{t_i t_j}\right)_{0 \leq t_i, t_j < l^2} \in \mathrm{Mat}_{l^2 \times l^2}(\mathbb{F}_q)$. 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 $\widetilde{F_i}$ including the oil variable X_k is of the form

$$\sum_{\alpha=0}^{l^2+l-1} A_{i,\alpha} \cdot \left(\sum_{j=0}^{v-1} X_j^t \left(Q_{i,\alpha,1} F_{i',jk} Q_{i,\alpha,2}\right) X_k \right) \cdot B_{i,\alpha}$$
$$+ \sum_{\alpha=0}^{l^2+l-1} A_{i,\alpha} \cdot \left(\sum_{j=0}^{v-1} X_k^t \left(Q_{i,\alpha,1} F_{i',kj} Q_{i,\alpha,2}\right) X_j \right) \cdot B_{i,\alpha} \tag{1.3}$$

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 $\widetilde{F_i}$ as follows.

Algorithm 13: 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 $(A_{i,\alpha}, B_{i,\alpha}, Q_{i,\alpha,1}, Q_{i,\alpha,2}$ for $0 \leq \alpha < l^2 + l)$
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 $i \leftarrow 0$ **to** $m - 1$ **do**
  for $\alpha \leftarrow 0$ **to** $l^2 + l - 1$ **do**
    for $j \leftarrow 0$ **to** $v - 1$ **do**
      $\mathbf{Left}_{i,\alpha}[j] \leftarrow A_{i,\alpha} * X_j^t * Q_{i,\alpha,1}$ ▷ the left term of $F_{i,jk}$
      $\mathbf{Right}_{i,\alpha}[j] \leftarrow Q_{i,\alpha,2} * X_j * B_{i,\alpha}$ ▷ the right term of $F_{i,jk}$
    end
  end
end
for $t_i \leftarrow 0$ **to** $l^2 + l - 1$ **do**
  for $t_j \leftarrow 0$ **to** $l^2 + l - 1$ **do**
    $M_{ik}[t_i][t_j] \leftarrow 0$
  end
end
for $i \leftarrow 0$ **to** $m - 1$ **do**
  for $\alpha \leftarrow 0$ **to** $l^2 + l - 1$ **do**
    $i' \leftarrow (i + \alpha) \mod m$
    for $j \leftarrow 0$ **to** $v - 1$ **do**
      $\mathbf{Left}_{X_k} \leftarrow \mathbf{Left}_{i,\alpha}[j] * F_{i'}^{12}[j][k] * Q_{i,\alpha,2}$ ▷ the left term of X_k
      $\mathbf{Right}_{X_k} \leftarrow B_{i,\alpha}$ ▷ the right term of X_k
      for $t_i \leftarrow 0$ **to** $l^2 + l - 1$ **do**
        for $t_j \leftarrow 0$ **to** $l^2 + l - 1$ **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
    for j *for* 0 *to* $v - 1$ **do**
      $\mathbf{Left}_{X_k} \leftarrow A_{i,\alpha}$ ▷ the left term of X_k^t
      $\mathbf{Right}_{X_k} \leftarrow Q_{i,\alpha,1} * F_{i'}^{21}[k][j] * \mathbf{Right}_{i,\alpha}[j]$ ▷ the right term of X_k^t
      for $t_i \leftarrow 0$ **to** $l^2 + l - 1$ **do**
        for $t_j \leftarrow 0$ **to** $l^2 + l - 1$ **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
end
return M_{ik}

We are now ready to write down the linear system of the oil variables over the finite field $\mathbb{F}_q$. We put the coefficient matrices M_{ik} and the vinegar part values $\widetilde{F}_{i,VV}$ of the central map into the augmented matrix G of the system as follows.

Algorithm 14: 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 **digest** $= Hash(D)$
length of the digest $|\textbf{digest}|$
public seed $\mathbf{s_{public}}$
salt

output: the augmented matrix G

$m \leftarrow o$
$(G[0][m * l^2], \ldots, G[m * l^2 + l - 1][m * l^2]) \leftarrow$
  $Hash_{SHAKE256}(\mathbf{s_{public}}||\textbf{digest}||\textbf{salt})$
    ▷ Put the hash value in the last column of G
for $i \leftarrow 0$ **to** $m - 1$ **do**
  for $j \leftarrow 0$ **to** $l - 1$ **do**
    for $k \leftarrow 0$ **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 $i \leftarrow 0$ **to** $m - 1$ **do**
  for $k \leftarrow 0$ **to** $m - 1$ **do**
    for $t_i \leftarrow 0$ **to** $l^2 + l - 1$ **do**
      for $t_j \leftarrow 0$ **to** $l^2 + l - 1$ **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 **flag_redo** $\in \{\textbf{TRUE}, \textbf{FALSE}\}$ 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 15: 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 $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \le \alpha < l^2 + l)$ using Algorithm 7
$m \leftarrow o$
Generate $(T^{12}, F_i^{11}, F_i^{12}, F_i^{21}$ for $0 \le i < m)$ using Algorithm 10
$[T^{-1}] \leftarrow \begin{bmatrix} I^{11} & T^{12} \\ 0 & I^{22} \end{bmatrix}$
$\mathbf{num_{sig}} \leftarrow 0$
repeat
    $\mathbf{num_{sig}} \leftarrow \mathbf{num_{sig}} + 1$
    Assign vinegar values $(X_0, \ldots, X_{v-1})$ using Algorithm 11
    Compute $(F_{i,VV}$ for $0 \le i < m)$ using Algorithm 12
    Compute $(M_{ik}$ for $0 \le i < m$, $0 \le k < o)$ using Algorithm 13
    Build the augmented matrix G using Algorithm 14
    flag_redo, $(\widetilde{X}_0, \widetilde{X}_1, \ldots, \widetilde{X}_{o-1}) \leftarrow$ **Gauss**(G)
    If $q \neq 16$: Set **flag_redo** $\leftarrow$**TRUE** if any of the $(\widetilde{X}_0, \widetilde{X}_1, \ldots, \widetilde{X}_{o-1})$ is symmetric
until flag_redo==FALSE;
sig $\leftarrow [T^{-1}](X_0, \ldots, X_{v-1}, \widetilde{X}_0, \ldots, \widetilde{X}_{o-1})^t$
return (**sig**, **salt**) as bytes

1.9.4 Algorithms for Signature Verification

Recall that the public key $\widetilde{P} = \left[\widetilde{P}_0, \cdots, \widetilde{P}_{m-1}\right] : \mathcal{R}^n \to \mathcal{R}^m$, where

$$\widetilde{P}_i(\mathbf{U}) = \sum_{\alpha=0}^{l^2+l-1} \sum_{d_j=0}^{n-1} \sum_{d_k=0}^{n-1} A_{i,\alpha} \cdot U_{d_j}^t (Q_{i,\alpha,1} P_{i',d_j d_k} Q_{i,\alpha,2}) U_{d_k} \cdot B_{i,\alpha}$$

with the variable $\mathbf{U} = (U_0, \ldots, U_{m-1})^t$. 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 $0 \le i < m$. The algorithm for evaluating the public map at a signature **sig** is as follows.

Algorithm 16: Evaluate the public map

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

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

```
m ← o
for α ← 0 to m − 1 do
    for j ← 0 to n − 1 do
        Left_{i,α}[j] ← A_{i,α} * (sig[j])^t * Q_{i,α,1}        ▷ the left term of P_{i,d_j d_k}
        Right_{i,α}[j] ← Q_{i,α,2} * sig[j] * B_{i,α}           ▷ the right term of P_{i,d_j d_k}
    end
end
for i ← 0 to m − 1 do
    i' ← (i + α) mod m
    hash_s[i] ← 0
    for α ← 0 to l^2 + l − 1 do
        for d_j ← 0 to v − 1 do
            for d_k ← 0 to v − 1 do
                hash_s[i] = hash_s[i] + Left_{i,α}[d_j] * P^{11}_{i'}[d_j][d_k] * Right_{i,α}[d_k]
            end
        end
        for d_j ← 0 to v − 1 do
            for d_k ← 0 to o − 1 do
                hash_s[i] = hash_s[i] + Left_{i,α}[d_j] * P^{12}_{i'}[d_j][d_k] * Right_{i,α}[v + d_k]
            end
        end
        for d_j ← 0 to o − 1 do
            for d_k ← 0 to v − 1 do
                hash_s[i] = hash_s[i] + Left_{i,α}[v + d_j] * P^{21}_{i'}[d_j][d_k] * Right_{i,α}[d_k]
            end
        end
        for d_j ← 0 to o − 1 do
            for d_k ← 0 to o − 1 do
                hash_s[i] = hash_s[i] + Left_{i,α}[v + d_j] * P^{22}_{i'}[d_j][d_k] * Right_{i,α}[v + d_k]
            end
        end
    end
end
hash_s ← (hash_s[0], ..., hash_s[m − 1])^t
return hash_s
```

Algorithm 17: Signature verification

input : SNOVA parameters (v, o, l)
public key $(\mathbf{s_{public}}, P^{22}_{i'}$ for $0 \le i < m)$
digest of the document **digest** $= Hash(D)$
length of the digest —**digest**—
the signature **sig** and **salt**

output: Accept or **Reject**

return Reject if any of the matrices of **sig** is symmetric
Generate $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $0 \le \alpha < l^2 + l$ using Algorithm 7
$m \leftarrow o$
Generate $(P^{11}_{i'}, P^{12}_{i'}, P^{21}_{i'}$ for $0 \le i < m)$ using Algorithm 7
$\mathbf{hash_d} \leftarrow Hash_{SHAKE256}(\mathbf{s_{public}} || \mathbf{digest} || \mathbf{salt})$
Compute $\mathbf{hash_s}$ using Algorithm 16
if $\mathbf{hash_s} == \mathbf{hash_d}$ **then**
  return Accept
else
  return Reject
end

1.10 Parameters Settings

In this section, we propose our parameters aiming at three security levels in the new call of NIST PQC project [44] levels I, III and V, respectively.

1.10.1 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 included in the size of SNOVA signature (in order to attain EUF-CMA) and the 16 bytes seed is included in the size of public key size.

Table 8: Key-sizes and lengths of the signature of SNOVA parameter settings (in bytes).

Security Level	(v, o, q, l)	Size_{pk}	Size_{sig}	$\text{Size}_{\text{esk}}/\text{Size}_{\text{ssk}}$
I	$(37, 17, 16, 2)$	9842	124	91440/48
	$(25, 8, 16, 3)$	2320	164.5	39576/48
	$(24, 5, 16, 4)$	1016	248	36848/48
III	$(56, 25, 16, 2)$	31266	178	300848/48
	$(49, 11, 16, 3)$	6005.5	286	177060/48
	$(37, 8, 16, 4)$	4112	376	133040/48
	$(24, 5, 16, 5)$	1578.5	378.5	60048/48
V	$(75, 33, 16, 2)$	71890	232	704532/48
	$(66, 15, 16, 3)$	15203.5	380.5	435423/48
	$(60, 10, 16, 4)$	8016	576	395248/48
	$(29, 6, 16, 5)$	2716	453.5	100398/48

Table 9: Preliminary key-sizes and lengths of the signature of odd q SNOVA parameter settings (in bytes).

Security Level	(v, o, q, l)	Size_{pk}	Size_{sig}	Size_{ssk}
I	$(24, 5, 23, 4)$	616	282	48
III	$(37, 8, 19, 4)$	2269	400	48
V	$(60, 10, 23, 4)$	4702	656	48

Remark 1.1. During Round 1 evaluation, we discussed the MinRank problem induced by ring UOV in our Round 1 supporting document. The decision regarding the number of vinegar variables in our Round 1 parameter sets was primarily based on the discussion of the MinRank of quadratic forms induced by ring UOV. In [34], it is shown that the solutions of such MinRank problems are not useful to an attacker. Therefore, we believe that there is room for adjustment and reduction in the number of vinegar variables. The security analysis of a reduction of vinegar variables in our parameter sets is not mature enough to submit these for Round 2. We have therefore decided to stay with the existing parameters.

1.10.2 How the Performance is 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. For a fixed security level, by increasing the parameter l, we can further reduce the value of m. Therefore, for larger l, we will have smaller public key size. On the other hand, a larger 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 multiple parameter sets 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 Time

The official implementation of SNOVA [54] supports two optimization levels, an optimized version that uses plain C and a version that uses AVX2 to vectorize the matrix multiplications if $q = 16, l \neq 4$.

Table 10: Benchmark results on a modern desktop system (Arrow Lake) using AVX2 instructions. The performance is the median number of CPU cycles over 2048 benchmark runs.

SL	(v, o, q, l)	XOF	KeyGen ESK	Sign SSK	Sign ESK	Verify
I	(37, 17, 16, 2)	AES	754,677	737,085	262,422	151,842
	(37, 17, 16, 2)	SHAKE	923,342	905,717	259,619	317,564
	(25, 8, 16, 3)	AES	246,298	494,904	340,740	176,153
	(25, 8, 16, 3)	SHAKE	317,078	563,468	346,892	244,364
	(24, 5, 16, 4)	AES	222,876	530,409	385,429	205,829
	(24, 5, 16, 4)	SHAKE	286,892	594,690	385,965	268,342
III	(56, 25, 16, 2)	AES	3,289,256	2,876,627	648,245	531,159
	(56, 25, 16, 2)	SHAKE	3,829,031	3,424,697	651,370	1,079,757
	(49, 11, 16, 3)	AES	1,503,852	2,061,825	984,405	675,680
	(49, 11, 16, 3)	SHAKE	1,839,992	2,390,210	997,315	987,091
	(37, 8, 16, 4)	AES	1,039,795	1,804,955	1,102,301	652,463
	(37, 8, 16, 4)	SHAKE	1,263,702	2,030,093	1,100,166	889,065
	(24, 5, 16, 5)	AES	456,998	1,421,115	1,120,762	685,307
	(24, 5, 16, 5)	SHAKE	553,744	1,524,408	1,128,682	787,049
V	(75, 33, 16, 2)	AES	9,745,037	8,052,855	1,426,465	1,309,034
	(75, 33, 16, 2)	SHAKE	11,034,489	9,328,179	1,450,322	2,595,930
	(66, 15, 16, 3)	AES	4,561,008	5,700,004	2,215,902	1,737,533
	(66, 15, 16, 3)	SHAKE	5,376,257	6,516,854	2,223,800	2,567,815
	(60, 10, 16, 4)	AES	3,519,404	5,574,943	2,800,653	1,932,041
	(60, 10, 16, 4)	SHAKE	4,244,420	6,216,789	2,819,456	2,648,472
	(29, 6, 16, 5)	AES	804,402	2,278,333	1,758,416	1,073,040
	(29, 6, 16, 5)	SHAKE	985,666	2,442,798	1,757,620	1,249,950

The benchmarks are for the official implementation of SNOVA [54] using AVX2 instructions. The performance is reported in CPU cycles. The number of cycles is the median over 2048 benchmark runs. Test system: CPU Intel(R) Core(TM) Ultra 7 265K (Arrow Lake), Compiler: GCC 14.2.0, OS: Linux Mint 22, Turbo Boost: disabled.

Table 11: Benchmark results on a laptop (Intel(R) Core(TM) Ultra 7 155H (Meteor Lake), compiler: gcc 15.2.1) for the optimized version. The performance is the median number of CPU cycles over 2048 benchmark runs. For comparison we included the results on the same platform of the AVX2 optimized version as submitted to NIST Round 2 ("R2").

SL	(v, o, q, l)	XOF	KeyGen SSK	Sign SSK	Verify
I	(24, 5, 23, 4)	AES	306,600	618,634	212,794
	(24, 5, 16, 4)	AES	258,402	751,578	189,870
	R2(24, 5, 16, 4)	AES	371,054	839,914	317,134
	(24, 5, 23, 4)	SHAKE	455,498	759,376	356,810
	(24, 5, 16, 4)	SHAKE	367,092	859,515	296,166
	R2(24, 5, 16, 4)	SHAKE	486,749	964,477	430,798
III	(37, 8, 19, 4)	AES	1,503,369	2,424,519	674,507
	(37, 8, 16, 4)	AES	1,414,838	3,239,046	619,674
	R2(37, 8, 16, 4)	AES	1,627,368	2,882,394	1,091,840
	(37, 8, 19, 4)	SHAKE	2,048,702	2,959,170	1,199,574
	(37, 8, 16, 4)	SHAKE	1,825,345	3,672,860	1,012,462
	R2(37, 8, 16, 4)	SHAKE	2,176,692	3,380,944	1,529,565
V	(60, 10, 23, 4)	AES	5,284,875	6,583,659	1,895,773
	(60, 10, 16, 4)	AES	5,501,406	9,999,327	1,868,829
	R2(60, 10, 16, 4)	AES	7,136,373	10,135,882	3,249,581
	(60, 10, 23, 4)	SHAKE	6,841,373	8,363,763	3,490,723
	(60, 10, 16, 4)	SHAKE	6,732,357	11,501,644	3,059,734
	R2(60, 10, 16, 4)	SHAKE	8,517,743	11,923,319	4,579,079

2.2 Space

The key sizes are given by the following expressions:

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

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

bytes.

Expanded private key size. The size of private key is

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

bytes.

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

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

bytes.

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

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

bytes.

3 Known Answer Test values (2.B.3)

The submission includes Known Answer Tests (KAT) files of the SNOVA signature scheme. We have provided scripts to generate the KAT files in the submission package. The SNOVA KAT files can also be downloaded at https://github.com/PQCLAB-SNOVA/SNOVA_KAT. This download link for our KAT files can be found in the submission folder KAT.

4 Expected Security Strength (2.B.4)

Note: This chapter is the unchanged 2.0 version. It needs to be updated.

We give the expected security strength of our parameters aiming at three security levels in the new call of the NIST PQC project [44] levels I, III, and V, respectively.

4.1 Security Strength

The complexity estimations of SNOVA parameter sets against the known attacks in Section 5 are shown in the following tables. The lowest complexity among all known attacks is marked in bold fonts.

Table 12: Complexity estimates of SNOVA parameter sets against forgery attacks in $\log_2(\sharp\text{gates})$.

SL	(v, o, q, l)	Direct	Collision	MLC	Forgery 1	Forgery 2
I	$(37, 17, 16, 2)$	165	168	158	188	167
	$(25, 8, 16, 3)$	171	176	**166**	217	171
	$(24, 5, 16, 4)$	184	192	182	219	**180**
III	$(56, 25, 16, 2)$	234	248	222	264	236
	$(49, 11, 16, 3)$	226	244	**220**	249	228
	$(37, 8, 16, 4)$	287	360	**279**	361	281
	$(24, 5, 16, 5)$	281	348	273	435	$\approx 281^*$
V	$(75, 33, 16, 2)$	302	376	**287**	344	303
	$(66, 15, 16, 3)$	302	388	**293**	333	303
	$(60, 10, 16, 4)$	350	488	**343**	392	347
	$(29, 6, 16, 5)$	334	448	323	517	$\approx 334^*$

In Table 12, “Direct”, “Collision”, “MLC”, “Forgery 1” and “Forgery 2” denote the Direct attack with Hashimoto’s algorithm in Section 5.2.1, the Collision attack in Section 5.2.2, the Memoryless Collision attack in Section 5.2.2, the Forgery attack proposed by Beullens in Section 5.2.3 and Forgery attack proposed by Cabarcas *et al.* in Section 5.2.4, respectively.

Our current methods do not enable us to provide an accurate estimate for the $l = 5$ parameter sets (indicated by a $*$ in Table 12). We can observe that the complexities of the Direct attack and the Forgery 2 attack are quite close, with the largest difference being only 6. Notably, in all our parameter sets, the ratio between the number of vinegar variables and oil variables is consistent. Therefore, we heuristically expect that for the parameter set with $l = 5$, the complexities of these two attacks will not differ significantly. We expect that the $l = 5$ parameter sets satisfy the claimed security levels with a wide margin.

Table 13: Complexity estimates of SNOVA parameter sets against key recovery attacks in $\log_2(\sharp\text{gates})$.The numbers with "$*$" depend on the heuristic prediction, for more details, we refer to Section 5.3.7.

SL	(v, o, q, l)	Rec. 1	Rec. 2	KS	Int.
I	$(37, 17, 16, 2)$	203	195	165	**153**
	$(25, 8, 16, 3)$	200	187	209	221
	$(24, 5, 16, 4)$	269	249	309	353
III	$(56, 25, 16, 2)$	297	288	253	**221**
	$(49, 11, 16, 3)$	438	424	461	529
	$(37, 8, 16, 4)$	387	365	469	506
	$(24, 5, 16, 5)$	281	**257***	385	412
V	$(75, 33, 16, 2)$	389	379	341	288
	$(66, 15, 16, 3)$	574	558	617	690
	$(60, 10, 16, 4)$	695	673	805	922
	$(29, 6, 16, 5)$	334	**310***	465	494

In Table 13, "Rec. 1", "Rec. 2", "KS" and "Int." denote the Reconciliation attack in Section 5.3.1, the Reconciliation attack proposed by Cabarcas *et al.* in Section 5.3.7, the Kipnis-Shamir attack in Section 5.3.2 and Intersection attack in Section 5.3.3, respectively.

Table 14: Complexity estimates of SNOVA parameter sets against key recovery attacks via lifting method in $\log_2(\sharp\text{gates})$.

SL	(v, o, q, l)	Lifting Rec.	Lifting KS	Lifting Int.
I	$(37, 17, 16, 2)$	196	167	157
	$(25, 8, 16, 3)$	195	212	275
	$(24, 5, 16, 4)$	319	313	510
III	$(56, 25, 16, 2)$	289	255	289
	$(49, 11, 16, 3)$	510	464	510
	$(37, 8, 16, 4)$	425	473	425
	$(24, 5, 16, 5)$	320	389	623
V	$(75, 33, 16, 2)$	376	343	302
	$(66, 15, 16, 3)$	655	620	918
	$(60, 10, 16, 4)$	785	809	1319
	$(29, 6, 16, 5)$	384	469	734

In Table 14, “Lifting Rec.”, “Lifting KS” and “Lifting Int.” denote the Lifting Reconciliation attack in Section 5.3.4, the Lifting Kipnis-Shamir attack in Section 5.3.5 and Lifting Intersection attack in Section 5.3.6, respectively.

5 Analysis of Known Attacks (2.B.5)

Note: This chapter is the unchanged 2.0 version. It needs to be updated.

The SNOVA scheme can be considered as both a UOV-like signature scheme over the matrix ring $\mathcal{R}$ and a UOV over $\mathbb{F}_q$. In the Round 1 of evaluation, several attacks were proposed [34, 37, 11, 16, 41, 2]. These attacks are mainly based on the structure of $\mathbb{F}_q[S]$ and the low rank possibility of public map under bilinear formulation [65]. In this section, we will review these attacks and analyze their impact following the current tweaks on SNOVA, as well as evaluate their complexity. We hope that through a comprehensive analysis, the structure of SNOVA can be better understood. At the same time, we demonstrate that with these Round 2 modifications, SNOVA not only maintains its performance but also becomes more secure. We would like to emphasize that, with slight tweaks, SNOVA continues to meet the security requirement of NIST when facing these attacks.

Forgery attacks. Finding the preimage of the public map for the hash value of a message is what constitutes signature forgery.

In [34], a forgery attack targeting ring UOV has been proposed. However, as mentioned in the same paper, the public map 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 $\widetilde{F}_i$ of SNOVA. The forgery attack in [34] targeting the ring UOV can not be applied to SNOVA.

In [11], Beullens proposed a forgery attack against SNOVA, pointing out that the Round 1 SNOVA public map may have a low-rank issue. Beullens interprets the public map of SNOVA with bilinear form formulation. Under this formulation, he discovered that if the $A_\alpha, B_\alpha, Q_{\alpha,1}, Q_{\alpha,2}$ matrices in SNOVA are generated randomly then the matrices $\mathbf{E}_{j,k}$ in the formulation may have a low-rank linear combination and this gives a forgery attack. The attack shows that the matrix $\mathbf{E}_{j,k} \in \mathbb{F}_q^{ml^2 \times ml^2}$ is block diagonal matrix with m identical blocks of size $l^2 \times l^2$ on the diagonals. Moreover, these $\mathbf{E}_{j,k}$ matrices are determined by the $A_\alpha, B_\alpha, Q_{\alpha,1}, Q_{\alpha,2}$ matrices. In his paper, Beullens also mentioned that with some slight adjustments his attacks can be prevented. We carefully analyzed his attack as well as the structure of $\mathbf{E}_{j,k}$ in SNOVA. We found other adjustments that we believe to be better. Under the Round 2 adjustments, $\mathbf{E}_{j,k}$ is no longer a block diagonal matrix and will become more like a random matrix. We will also explain that, in this case, the occurrence of low-rank cases will be highly unlikely.

In [16], the authors proposed a forgery attack based on a bilinear formulation. This attack primarily leverages the structure of $\mathbb{F}_q[S]$ and the techniques of stable ideals.This forgery attack is similar to Beullens' attack, as both require the SNOVA public map to be low rank. In other words, if the SNOVA public map is not of low rank, their attacks become inefficient. Both attacks rely on the low-rank property.

We will demonstrate that, under the current Round 2 adjustments, it is highly unlikely for the SNOVA public key map to be of low rank. With the adjustments made for Round 2, SNOVA continues to meet NIST security requirements against all known attacks. Finally, we will observe that our Round 2 tweaks do not increase the sizes of the public keys or signatures.

Key recovery attacks. Note that the public keys of SNOVA are generated by the congruence relation $[P_i] = [T]^t [F_i] [T]$. Since $[P_i] \in \mathrm{Mat}_{n\times n}(\mathcal{R})$ and $\mathrm{Mat}_{n\times n}(\mathcal{R}) \cong \mathrm{Mat}_{ln\times ln}(\mathbb{F}_q)$, for key recovery attacks, the security of SNOVA will be evaluated by analyzing the complexity of such attacks against the (lv, lo, q)-UOV induced from public key $[P_i], \ldots, [P_m]$.

5.1 Preliminaries

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

Solving MQ problem. The complexity of solving M homogeneous quadratic equations in N variables [17] can be estimated by

$$MQ(N, M, q) = 3 \cdot \binom{N - 1 + d_{reg}}{d_{reg}}^2 \cdot \binom{N+1}{2} \tag{5.1}$$

field multiplications. The term d_{reg}, degree of regularity of a semi-regular polynomial system [6], equals the smallest positive integer d such that the coefficient of t^d term in the series generated by

$$\frac{(1-t^2)^M}{(1-t)^N} \tag{5.2}$$

is non-positive.

Hybrid approach. The hybrid approach [7] randomly guesses k variables before solving the MQ system and the corresponding complexity is

$$MQ_{\mathsf{Hybrid}}(N, M, q) = q^k \cdot MQ(N - k, M, q) \tag{5.3}$$

field multiplications for the classical case and

$$q^{k/2} \cdot MQ(N - k, M, q) \tag{5.4}$$

field multiplications when applying Grover's algorithm [31] for the quantum case.

Methods for solving underdetermined MQ. On the other hand, several methods [59, 29, 32] have been proposed to solve underdetermined MQ more efficiently.

These methods can transform an underdetermined $MQ(N,\ M,\ q)$ problem to an $MQ(M-k-\alpha_k,\ M-\alpha_k,\ q)$ problem where the value of α_k depends on the approach utilized in each method. (Generally, the attack in [32] would be the sharpest among [59, 29, 32].) Hence, the main term of complexity of solving MQ system under this technique is given by

$$\min_k q^k \cdot MQ(M-k-\alpha_k, M-\alpha_k, q) \tag{5.5}$$

field multiplications in the classical case.

Recently, the algorithm in [32] has been revised. The updated algorithm has become more efficient. The complexity estimation formula is

$$MQ_{\mathsf{Hashimoto}}(N, M, q) \tag{5.6}$$
$$=(M-\alpha-k+1)MQ_{\mathsf{Hybrid}}(\alpha, \alpha, q) \tag{5.7}$$
$$+\, q^k \left(MQ_{\mathsf{Hybrid}}(\alpha-1, \alpha-1, q) + MQ_{\mathsf{Hybrid}}(M-\alpha-k, M-\alpha, q)\right). \tag{5.8}$$

provided that $N \geq \max\{(\alpha+1)(M-k-\alpha+1, \alpha(M-k)-(\alpha-1)^2+k\}$ holds.

Algorithms for super-underdetermined MQ. Note that, [36, 19, 40, 18] indicate that when the number of variables N is sufficiently larger than the number of equations M in an MQ problem then we can solve this MQ in polynomial time. Please refer to the table in [32] for more information. Note that these four algorithms are not applicable to the parameter settings of SNOVA.

5.2 Forgery Attacks

5.2.1 Direct Attack with Hashimoto's Algorithm

For a quadratic multivariate polynomial system $P = [P_1, \cdots, P_m]$ consisting of m equations in n variables over $\mathbb{F}_q$ and an intended $\mathbf{y} \in \mathbb{F}_q^m$, an attacker can directly try to solve the solution $\mathbf{u}$ of the system $P(\mathbf{u}) = \mathbf{y}$ algebraically with Gröbner basis approaches, such as those in [25, 26, 20, 17, 66]. We can assign values to $n-m$ variables in the system $P(\mathbf{u}) = \mathbf{y} = Hash(\mathbf{digest}||\mathbf{salt})$ randomly and then obtain an MQ system of m equations in n variables which can be solved with high probability. Once the system is solved, the solution $\mathbf{u}$ will be a valid fake signature that satisfies $P(\mathbf{u}) = \mathbf{y}$.

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

The complexity of classical direct attack is given by the estimation in [32]

$$\mathrm{Comp}_{\mathsf{Direct}} = MQ_{\mathsf{Hashimoto}}(l^2 n, l^2 m, q) \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \tag{5.9}$$

gates.

5.2.2 Collision Attack

To forge a fake signature, an attacker can also try to check M intended signatures $\mathbf{U}_j$ where $j = 1, \cdots, M$, and N hash values $Hash(\mathbf{digest}||\mathbf{salt}_k)$ where $k = 1, \cdots, N$, whether there exists a collision $P(\mathbf{U}_j) = Hash(\mathbf{digest}||\mathbf{salt}_k)$. And if it does, then the attacker has a valid fake signature. Thus, M signature computations and N hash value computations are involved. Therefore, according to the estimation of [14], the cost of such a collision attack would be

$$M \cdot (l^2 m) \cdot \left(2(\log_2 q)^2 + 3 \cdot \log_2 q\right) + N \cdot 2^{17} \tag{5.10}$$

gates in the sense that regarding SNOVA as a UOV scheme over $\mathbb{F}_q$. A collision is to be expected when $MN \approx q^{l^2 m}$. In a straightforward optimization of equation 5.10, the optimal value occurs when $N \approx q^{l^2 m/2}$ (note that at optimal values $M > N$). At SL III this optimal value of N requires a storage capacity of the order of 2^{192} bits which is more than the number of atoms in the Earth ($\approx 2^{167}$). Limiting M to the number of atoms in the Earth gives a cost of

$$\mathrm{Comp}_{\mathsf{Collision}} = \frac{q^{l^2 m}}{2^{167}} \cdot 2^{17}. \tag{5.11}$$

This is an unavoidable lower bound for the complexity of the collision attack when $q^{l^2 m/2} > 2^{167}$, as is the case for all parameters at SL III and SL V.

Memory-Time tradeoff. In the call for proposals [44], NIST has indicated that a metric to consider is the cost of accessing extremely large amounts of memory. For the collision attack a more basic aspect has to be taken into account: the cost of actually obtaining the amount of storage needed by the attack. For the proposed SNOVA parameters, the lowest value of $l^2 m$ is for $(37, 17, 16, 2)$ where $l^2 m \log_2(q) = 272$. In this case the optimal value for equation 5.10 is attained when $M \approx 2^{138}$ and $N \approx 2^{134}$. These are not the cost-optimal values however. Suppose that the collision attack runs at 1GHz and has a runtime of about a year. As there are about $3 \cdot 10^{16}$ cycles in a year, the cost of a $N \approx 2^{134}$ bits of storage has to be comparable to the cost of 2^{96} computational units for the given values of N and M to be cost-optimal. This is not realistic. For our estimate of the collision attack we have assumed a runtime of a year and we have assumed that the cost of a single bit of (persistent) storage is cheaper than a single logic gate by a factor of 10^6. When this cost factor is taken into account, the cost estimate of the collision attack becomes

$$\mathrm{Comp}_{\mathsf{Collision}} = q^{l^2 m/2} \cdot 2\left((l^2 m)\left(2(\log_2 q)^2 + 3 \cdot \log_2 q\right) \cdot 2^{17} \cdot 2^{35}\right)^{1/2}, \tag{5.12}$$

gates. Equation 5.12 can only apply when $q^{l^2m/2} < 2^{167}$ as explained above.

Taking into account the cost of memory makes the traditional collision attack less economical than the memoryless collision attack described below. This conclusion is largely independent of the specifics of the assumptions. Changing the runtime to a weekend and changing the relative cost of a storage bit versus one unit of computation to a very unrealistic value of 10^9 does not affect the outcome: The memoryless collision attack is more economical than a traditional collision attack.

For SNOVA SL I we have presented the numbers taking into account the estimated cost of storage, equation 5.12. At SL III and SL V we have used equation 5.11. Note that all SNOVA schemes satisfy the required complexities even if the cost of obtaining and accessing the memory is ignored.

Memoryless Collision Attack. The collision attack comes with a huge memory requirement. Memoryless collision-finding algorithm are considered to be more realistic than a traditional collision attack. As far as we know, for SNOVA the attack of Nikolic and Sasaki [42] is the most efficient memoryless collision attack currently available. This algorithm operates by producing a large number n_v of values v_i by iterative application of the public map $v_i = \tilde{P}(v_{i-1})$ and doing the same for a large number n_w of iterations on the hash function, $w_i = H(w_{i-1})$. The iterations over v_i and w_i will have a collision with high probability when $n_v n_w > q^{l^2m}$. For a complete description, we refer to [42].

Let N_P and $N_H = 2^{17}$ denote the complexities of calculating the public map and the hash (SHAKE256) respectively. The complexity of the algorithm of Nikolic and Sasaki [42] at time-optimal parameters is

$$\mathrm{Comp}_{\mathsf{MLC}} = q^{l^2m/2} \cdot 2\,(N_P N_H)^{1/2}\,. \tag{5.13}$$

The algorithm of Nikolic and Sasaki [42] has a small overhead that amounts to an additional factor between 1 and 2 in equation 5.13. We have ignored this overhead in our estimates. The number of hashes to be stored is

$$\frac{\max(N_P, N_H)}{\min(N_P, N_H)},$$

which we have also ignored as well as it is small. The Gray-code enumeration optimization [14] underlying the cost estimate 5.12 is very efficient for related inputs. It is not an efficient algorithm to produce the iterated values v_i, a direct evaluation of the public map $\tilde{P}(v_{i-1})$ is faster. Evaluating $\tilde{P}(v_{i-1})$ for arbitrary v_{i-1} has a complexity of slightly more than

$$N_P = mn^2l^3(l^2 + l)(2\log_2(q)^2 + \log_2(q)).$$

The vinegar variables can be set to some fixed value in the iteration. This reduces the complexity of the evaluation of $\tilde{P}(v_{i-1})$ by a factor n/m.

5.2.3 Forgery Attack Proposed by Beullens

We first describe the Beullens forgery attack against Round 1 SNOVA public map. SNOVA public map is a multivariate quadratic map characterized with $l \times l$ matrix ring $\mathcal{R} = \mathrm{Mat}_{l\times l}(\mathbb{F}_q)$ and a symmetric matrix S with irreducible characteristic polynomial. Beullens attack utilizes the low-rank possibility of SNOVA public map under the bilinear formulation. In the following, we adapt the notation in [11].

Bilinear formulation of Round 1 SNOVA public map. For a matrix $\mathbf{A}$ and a positive integer n, $\mathbf{A}^{\otimes n}$ denotes the block diagonal matrix with n copies of A on the block diagonal. For Round 1 SNOVA public map, $\widetilde{P}(\mathbf{U}) = \left(\widetilde{P}_1(\mathbf{U}), \ldots, \widetilde{P}_m(\mathbf{U})\right)$: $\mathcal{R}^n \to \mathcal{R}^m$, it can be expressed as, for $i \in \{1, 2, \ldots, m\}$,

$$\widetilde{P}_i(\mathbf{U}) = \sum_{\alpha=0}^{l^2-1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_\alpha \cdot U_j^t (Q_{\alpha,1} P_{i,jk} Q_{\alpha,2}) U_k \cdot B_\alpha \tag{5.14}$$

$$= \sum_{\alpha=0}^{l^2-1} A_\alpha \cdot \mathbf{U}^t \cdot Q_{\alpha,1}^{\otimes n} \cdot [P_i] \cdot Q_{\alpha,2}^{\otimes n} \cdot \mathbf{U} \cdot B_\alpha \tag{5.15}$$

where $[P_i]$ are the public keys of SNOVA, $Q_{\alpha,1}^{\otimes n}$ is a $n \times n$ diagonal matrix over $\mathcal{R}$ with identical blocks $Q_{\alpha,1}$ and similarly for $Q_{\alpha,2}^{\otimes n}$. Here, the vector $\mathbf{U} = (U_1, \cdots, U_n)^t \in \mathcal{R}^n$ is a matrix of height nl and width l when we regard it as over $\mathbb{F}_q$.

Let $\mathcal{B} : \mathbb{F}_q^{ln} \times \mathbb{F}_q^{ln} \to \mathbb{F}_q^{l^2 m}$ be the bilinear map defined as

$$\mathcal{B}_{i,a,b}(\mathbf{u}, \mathbf{v}) := \mathcal{B}_i^{(a,b)}(\mathbf{u}, \mathbf{v}). \tag{5.16}$$

where, for $(a, b) \in \{0, \ldots, \ell - 1\}^2$,

$$B_i^{(a,b)} : \mathbb{F}_q^{n\ell} \times \mathbb{F}_q^{n\ell} \to \mathbb{F}_q, B_i^{(a,b)}(\mathbf{u}, \mathbf{v}) := \mathbf{u}^t \left(S^a\right)^{\otimes n} [P_i] \left(S^b\right)^{\otimes n} \mathbf{v}. \tag{5.17}$$

where $\mathbf{u}_j$ is the $(j+1)$-th column of $\mathbf{U}$ for $j \in \{0, \ldots, l-1\}$.

In [11, 65], it can be seen that

$$\widetilde{P}(\mathbf{U}) = \sum_{j=0}^{l-1} \sum_{k=0}^{l-1} \mathbf{E}_{j,k} \cdot \mathcal{B}(\mathbf{u}_j, \mathbf{u}_k). \tag{5.18}$$

Note that for each $P_i(\mathbf{U})$ in Round 1 SNOVA public map, only one public key $[P_i]$ is used and A_α, B_α, $Q_{\alpha,1}$ and $Q_{\alpha,2}$ are shared by all equation. Therefore, the matrix $\mathbf{E}_{j,k}$ in the bilinear formulation of $P(\mathbf{U})$ is a block diagonal matrix with identical blocks, i.e., $\mathbf{E}_{j,k} = \widetilde{\mathbf{E}}_{j,k}^{\otimes m}$, $\widetilde{\mathbf{E}}_{j,k}$ is an $l^2 \times l^2$ matrix determined by matrices $A_\alpha, B_\alpha, Q_{\alpha,1}, Q_{\alpha,2}$.

We then briefly describe the attack by Beullens. For other details, we refer to [11].

Attack. This attack attempts to forge a signature by solving for $\mathbf{U}$ satisfy that the columns $\mathbf{u}_j = a_j\mathbf{u}_0 + v_j$ where $v_j \in \mathbb{F}_q^{ln}$ is randomly chosen for all $j \in \{1, \dots, l-1\}$, for some $a_1, \dots, a_{l-1} \in \mathbb{F}_q$. Under the formulation (5.18), this implies that the quadratic part of public map $P(\mathbf{U})$ is $\mathbf{E}_\alpha \cdot \mathcal{B}(\mathbf{u}_1, \mathbf{u}_1)$ where

$$\mathbf{E}_\alpha = \sum_{j=0}^{l-1}\sum_{k=0}^{l-1} a_j a_k \mathbf{E}_{jk}. \tag{5.19}$$

The attack is divided into three steps:

- Since for the Round 1 $\mathbf{E}_{j,k} = \widetilde{\mathbf{E}}_{j,k}^{\otimes m}$, the linear combination $\mathbf{E}_\alpha$ is also a block diagonal matrix of size $l^2m \times l^2m$ with identical $l^2 \times l^2$ blocks on diagonal. Therefore, if the linear combination of matrices $\widetilde{\mathbf{E}}_{jk}$ has rank defect d then the corresponding linear combination $\mathbf{E}_\alpha$ will have rank defect md. This gives a generalized MinRank problem.
- Following with the first step, if $d = l^2 - r$ then we have $\mathrm{rank}(\mathbf{E}_\alpha) = mr$. Therefore, for an attacker who wants to forge a fake signature, the remaining is to solving an MQ system of mr equations in $nl - m(l^2 - r)$ variables.
- Using the structure of $\mathbb{F}_q[S]$, the generalized MinRank problem in the first step can be extended to a generalized MinRank problem with $l(l-1)$ variables that tries to find the low rank of $\mathbf{E}_\mathbf{R}$ which is the quadratic part of $P(\mathbf{U})$ under the setting that $\mathbf{u}_j = \mathbf{R}_j^{\otimes n}\mathbf{u}_0 + \mathbf{v}_j$ where $\mathbf{R}_j \in \mathbb{F}_q[S]$ for all $j \in \{1, \dots, l-1\}$. This will allow the attackers to find matrices with lower rank. Hence, the number of variables in step 2 can be further reduced. Then, the attack becomes more efficient. Let $\mathbf{u}_j = R_j^{\otimes n}\mathbf{u}_0 + v_j$ and $\mathbf{u}_k = R_k^{\otimes n}\mathbf{u}_0 + v_k$ as defined in Beullens paper. Write $R_j = \sum_{a'=0}^{l-1} c_{a'}^{(j)} S^{a'}$ and $R_k = \sum_{b'=0}^{l-1} c_{b'}^{(k)} S^{b'}$. Therefore, the quadratic part in $\mathbf{E}_\mathbf{R}$ attack is

  $$\sum_{jk} \mathbf{E}_{jk}\mathcal{B}(R_j^{\otimes n}\mathbf{u}_0, R_k^{\otimes n}\mathbf{u}_0) = \sum_{jk}\sum_{a'b'} c_{a'}^{(j)} c_{b'}^{(k)} \mathbf{E}_{jk}\mathcal{B}((S^{a'})^{\otimes n}\mathbf{u}_0, (S^{b'})^{\otimes n}\mathbf{u}_0)$$

  And according to our understanding of Beullens paper,

  $$\mathcal{B}((S^{a'})^{\otimes n}\mathbf{u}_0, (S^{b'})^{\otimes n}\mathbf{u}_0) = \mathbf{E}'_{a',b'}\mathcal{B}(\mathbf{u}_0, \mathbf{u}_0)$$

  where $\mathbf{E}'_{a',b'}$ is the (linear transformation) matrix cooresping to the linear relation between two bilinear map $\mathcal{B}((S^{a'})^{\otimes n}\mathbf{u}_0, (S^{b'})^{\otimes n}\mathbf{u}_0)$ and $\mathcal{B}(\mathbf{u}_0, \mathbf{u}_0)$. Same resoning can be found in Beullens paper [11]. Hence, the quadraic part above can be write as

  $$\sum_{jk}\sum_{a'b'} c_{a'}^{(j)} c_{b'}^{(k)} \mathbf{E}_{jka'b'}\mathcal{B}(\mathbf{u}_0, \mathbf{u}_0).$$

  where $\mathbf{E}_{jka'b'} = \mathbf{E}_{jk}\mathbf{E}'_{a',b'}$ Under our modeling and notation,

  $$\mathbf{E}_\mathbf{R} = \sum_{jk}\sum_{a'b'} c_{a'}^{(j)} c_{b'}^{(k)} \mathbf{E}_{jka'b'}$$

which is a block diagonal matrix whose entries are quadratic functions of the coefficients of $R_1, \dots, R_{l-1}$ (this coincides with Beullens paper). Note that the matrix $\mathbf{E}'_{a',b'}$ is block diagonal matrix and so $\mathbf{E}_{jk}$ in the Round 1 setting of SNOVA. Thus, $\mathbf{E_R}$ is a block diagonal matrix with identical blocks in Beullens attack.

The forgery attack of Beullens is determined by $mr = \text{rank}(\mathbf{E_R})$ and the complexity is the cost of solving the MQ system of mr equations in $nl - m(l^2 - r)$ variables. This MQ system can be very underdetermined if the rank defect d at step 1 becomes larger, or equivalently, the rank defect of $\mathbf{E_R} = \widetilde{\mathbf{E}}_{\mathbf{R}}^{\otimes m}$ becomes lager where $\widetilde{\mathbf{E}}_{\mathbf{R}}$ is an $l^2 \times l^2$ matrix induced by $A_\alpha, B_\alpha, Q_{\alpha,1}, Q_{\alpha,2}$ in SNOVA. We can observe that Beullens's attack is based on the rank defect of the matrix $\mathbf{E_R}$.

Analysis of the attack against Round 2 SNOVA public map. Similarly, the Round 2 SNOVA public map, $\widetilde{P}(\mathbf{U}) = \left(\widetilde{P}_1(\mathbf{U}), \dots, \widetilde{P}_m(\mathbf{U})\right) : \mathcal{R}^n \to \mathcal{R}^m$ can be expressed as, for $i \in \{1, 2, \dots, m\}$,

$$\widetilde{P}_i(\mathbf{U}) = \sum_{\alpha=0}^{l^2+l-1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{i,\alpha} \cdot U_j^t (Q_{i,\alpha,1} P_{i',jk} Q_{i,\alpha,2}) U_k \cdot B_{i,\alpha} \tag{5.20}$$

$$= \sum_{\alpha=0}^{l^2+l-1} A_{i,\alpha} \cdot \mathbf{U}^t \cdot Q_{i,\alpha,1}^{\otimes n} \cdot [P_{i'}] \cdot Q_{i,\alpha,2}^{\otimes n} \cdot \mathbf{U} \cdot B_{i,\alpha} \tag{5.21}$$

where $i' = i + \alpha \mod m$.

In Beullens' original attack against Round 1 SNOVA public map, the main reason that the rank of $\mathbf{E_R}$ decreases significantly is that $\mathbf{E_R}$ is a block diagonal matrix with identical diagonal blocks. However, for Round 2 SNOVA public map, we can observe that, for $P_i(\mathbf{U})$, multiple public key $[P_{i'}]$, $(\alpha = 0, \dots, l^2 + l - 1)$ are used. Furthermore, each equation no longer shares the same A, B, Q matrices. Therefore, the matrix $\mathbf{E_R}$ is no longer a block diagonal matrix with identical diagonal blocks. $\mathbf{E_R}$ becomes more like a random matrix. Following the estimation of Beullens, the complexity of SNOVA against Beullens attack is given by solving the MQ system of $R = \text{rank}(\mathbf{E_R})$ equations in $ln - l^2m + R$ variables.

The minimal rank of $\mathbf{E_R}$, R. In the cases of $l = 2, 3$, Beullens performed an exhaustive search for the minimal rank of $\mathbf{E_R}$. Our approach is similar: for Round 2 SNOVA with $l = 2, 3$, the computations are feasible. For all parameter sets with $l = 2, 3$, we fixed a specific set of ABQ matrices and used completely search to check the corresponding minimal rank of $\mathbf{E_R}$, thus avoiding the weak key issue. For the case of $l = 4$, the complete search is not feasible. For our $l = 4$ parameter sets, we generated the ABQ matrices randomly. Based on heuristic analysis inspired by Beullens [11], we believe that when $l = 4$, $\mathbf{E_R}$ behaves very similarly to a random matrix. The expected minimal rank in this case is $l^2o - l + 1$. Under these conditions, SNOVA is resistant to Beullens' attack and does not suffer from the weak key issue.

For a $l^2o \times l^2o$ random matrix, it has a rank $\leq l^2o - d$ with probability q^{-d^2}. Therefore, we expect the minimal achievable rank to typically be $l^2o - l + 1$. According to the heuristic, we expect the probability that a matrix of rank $l^2o - d$ or lower exists to be roughly

$$1 - \left(1 - q^{-d^2}\right)^{q^{l(l-1)}}.$$

Note that for $d \geq l$ this is well approximated by $2^{-4(d^2 - l^2 + l)}$. For $l = 4, 5$, the probability that a matrix with rank defect $d = 7$ exists to be $\leq 2^{-148}$. Note that, in the case of $d = 7$, all our $l = 4, 5$ parameter sets still meet the security requirements. For more details, we refer to Table 17. Therefore, we conclude that the probability of $l = 4, 5$ weak key is negligible in practice. We summarize the information in the following table.

Table 15: Minimal rank of $\mathbf{E_R}$ of Round 2 SNOVA. The numbers with "$*$" depend on the heuristic prediction. The numbers without "$*$" were determined by complete search.

SL	(v, o, q, l)	*ABQ* matrices	minimal rank of $\mathbf{E_R}$
I	$(37, 17, 16, 2)$	fixed by a seed	$l^2o - l + 1$
	$(25, 8, 16, 3)$	fixed by a seed	$l^2o - l + 1$
	$(24, 5, 16, 4)$	randomly generated	$l^2o - l + 1^*$
III	$(56, 25, 16, 2)$	fixed by a seed	$l^2o - l + 1$
	$(49, 11, 16, 3)$	fixed by a seed	$l^2o - l + 1$
	$(37, 8, 16, 4)$	randomly generated	$l^2o - l + 1^*$
	$(24, 5, 16, 5)$	randomly generated	$l^2o - l + 1^*$
V	$(75, 33, 16, 2)$	fixed by a seed	$l^2o - l + 1$
	$(66, 15, 16, 3)$	fixed by a seed	$l^2o - l + 1$
	$(60, 10, 16, 4)$	randomly generated	$l^2o - l + 1^*$
	$(29, 6, 16, 5)$	randomly generated	$l^2o - l + 1^*$

In Table 16, we demonstrated the impact of different values for the number of terms in the summation over α. Let n_α denotes the number of terms in the summation over α. We observed that when $n_\alpha = l^2 + l$, the MinRank distribution converges to the expected minimal rank. This explains why we increased the number of terms in the summation over α to $l^2 + l$. In this case, the MinRank distribution aligns with the completely random case.

For $l = 2$ the number of *ABQ* seeds was 1000, for $l = 3, o = 8$ the number of seeds was 100 and for $l = 3, o = 15$ the number of seeds was 10. The remaining parameters sets with $l \leq 3$ give very similar results. Based on these results as well as the underlying rank distribution we have selected $n_\alpha = l^2 + l$ for SNOVA Round 2.

Remark 5.1. For $l = 2, 3$, we fixed the seed to a well-chosen value results in a MinRank$(\mathbf{E_R}) = l^2o - l + 1$, as shown in Table 15. For higher l, we have not found any seed that yields a lower MinRank. This was to be expected as the probability of finding a seed with rank drop $d = 3$ when $l = 3$ is about 2^{-12} (see above).

Table 16: Distribution of the rank drop d depending on the number of terms n_α in the sum over α. The rank drop $d = l^2 o - \mathrm{MinRank}(\mathbf{E_R})$ is the difference between the maximum value of the MinRank of $\mathbf{E_R}$, which is $l^2 o$, and the actual MinRank.

	(37, 17, 16, 2)				(75, 33, 16, 2)	
d	$n_\alpha = 4$	$n_\alpha = 5$	$n_\alpha = 6$	$n_\alpha = 8$	$n_\alpha = 4$	$n_\alpha = 6$
0	0	0	0	0	0	0
1	0	820	990	996	0	989
2	7	171	10	4	0	11
3	92	9			0	
4	267				2	
5	311				24	
6	219				107	
7	82				174	
8	17				235	
9	5				228	
10					136	
11					64	
12					23	
13					6	
14					1	
	(25, 8, 16, 3)				(66, 15, 16, 3)	
d	$n_\alpha = 9$	$n_\alpha = 10$	$n_\alpha = 11$	$n_\alpha = 12$	$n_\alpha = 12$	
0	0	0	0	0	0	
1	0	0	0	0	0	
2	4	100	100	100	10	
3	63					
4	30					
5	3					

The complexity of Beullens forgery attack is determined by $R = \mathrm{rank}(\mathbf{E_R})$. According to Table 15 above, we estimate the complexity as

$$\mathrm{Comp}_{\mathsf{Forgery1}} = MQ_{\mathsf{Hashimoto}}(ln - l^2 m + R, R, q) \cdot (2 \cdot (\log_2 q)^2 + \log_2 q). \tag{5.22}$$

gates, for all parameter sets, $R = l^2 m - l + 1$.

5.2.4 Forgery Attack Proposed by Cabarcas *et al.*

We first describe the forgery attack in [16] against Round 1 SNOVA public map. In [16], the authors focus on the structure of the equation

$$\mathbf{u}^t \left(S^a\right)^{\otimes n} [P_i] \left(S^b\right)^{\otimes n} \mathbf{u},\ i \in \{1, \ldots, m\},\ a, b \in \{0, \ldots, l-1\}. \tag{5.23}$$

For any system where the quadratic part takes the above form, it is referred to as a SNOVA system. The authors propose a multi-XL-like algorithm to solve SNOVA systems, based on the observation that the ideal I, generated by the quadratic part of 5.23, is $\mathbb{F}_q[S]$-stable. By applying an appropriate change of variables to the ideal I, they obtain a stable ideal under the action of a cyclic diagonal group of matrices. This transformation results in a polynomial system with a multi-degree homogeneous structure, which their XL-like algorithm effectively leverages.

In [16], the authors demonstrate that their XL-like algorithm can also improve the forgery attack described in [11]. Unlike the approach in [11], their method relates the forgery of a document D to solving a SNOVA system while preserving the multidegree homogeneity of the lifted system.

We summarize their forgery attack in several steps:

- As in Beullens attack, the SNOVA public map can be written as

  $$\widetilde{P}(\mathbf{U}) = \sum_{j=0}^{l-1} \sum_{k=0}^{l-1} \mathbf{E}_{j,k} \cdot \mathcal{B}(\mathbf{u}_j, \mathbf{u}_k).$$

  Similarly, they define $\mathbf{u}_j = \mathbf{R}_j^{\otimes n} \mathbf{u}_0$ where $\mathbf{R}_j \in \mathbb{F}_q[S]$. Then, we will have

  $$\widetilde{P}(\mathbf{u}_0) = \mathbf{E}_{\mathbf{R}} \cdot \mathcal{B}(\mathbf{u}_0, \mathbf{u}_0)$$

  The matrix $\mathbf{E}_{\mathbf{R}}$ may have rank defect.

- Let $\text{rank}(\mathbf{E}_{\mathbf{R}}) = mr$. Then, for a target document D, the attacker samples a **salt** such that the vector $Hash(Hash(D)||\textbf{salt}) \in \mathbb{F}_q^{l^2 m}$ lies in the column space of $\mathbf{E}_{\mathbf{R}}$. The probability of this occurring is

  $$q^{mr - l^2 m} = \frac{q^{mr}}{q^{l^2 m}}$$

  . Therefore, the cost of this step is

  $$q^{l^2 m - mr} \cdot l^6.$$

- Let $\text{rank}(\mathbf{E}_{\mathbf{R}}) = mr$. The attacker can find a set of $p = l^2 m - mr$ linearly independent vectors $w_1, \ldots, w_p \in \mathbb{F}_q^{l^2 m}$ in the kernel of $\mathbf{E}_{\mathbf{R}}$. Then, the attacker obtains a forgery by solving the system

  $$\mathbf{w} = \mathcal{B}(\mathbf{u}_0, \mathbf{u}_0) + \sum_{i=1}^{p} y_i \mathbf{w}_i,$$

for $\mathbf{u}_0$ over $\mathbb{F}_q^{ln}$ and $y_1, \ldots, y_p \in \mathbb{F}_q$. In this case, we are working with the multi-degree $\mathbf{d}^* \in \mathbb{Z}_{\geq 0}^l$ such that

$$0 \geq [\mathbf{t}^{\mathbf{d}}]H(t_1, \ldots, t_l) + p \cdot \sum_{\mathbf{d} \leq \mathbf{d}^*, \mathbf{d} \neq \mathbf{d}^*} [\mathbf{t}^{\mathbf{d}}]H(t_1, \ldots, t_l),$$

where

$$H(t_1, \ldots, t_l) = \frac{\prod_{1 \leq i < j \leq l} (1 - t_i t_j)^{2o} \cdot \prod_{i=1}^{l} (1 - t_i^2)^o}{\prod_{i=1}^{l} (1 - t_i)^{ol - \frac{k}{l}}}.$$

The cost of this step is estimated by

$$\min_{p \leq k \leq l^2 o,\, l | k} q^{k-p} \cdot 3 \left(ol - \frac{k}{l} \right)^2 \cdot \left[\widetilde{\mathcal{M}}(\mathbf{d}^*) \right]^2 \cdot \left(2 \cdot (\log_2 q^l)^2 + \log_2 q^l \right),$$

with $\widetilde{\mathcal{M}}(\mathbf{d}^*) = \mathcal{M}(\mathbf{d}^*) + p \cdot \sum_{\mathbf{d} \leq \mathbf{d}^*, \mathbf{d} \neq \mathbf{d}^*} \mathcal{M}(\mathbf{d})$, where $\mathcal{M}(\mathbf{d})$ is the number of monomials of multi-degree exactly $\mathbf{d}$.

- Finally, The total cost of this forgery attack is estimated by the maximum of the above two steps.

Analysis of the attack against Round 2 SNOVA public map. Similar to Beullens' attack, their forgery attack is also greatly affected by the rank of the $\mathbf{E_R}$ matrix. Under the Round 2 SNOVA public map, the $\mathbf{E_R}$ matrix is no longer a block diagonal matrix with identical diagonal blocks. $\mathbf{E_R}$ is now a $l^2 m \times l^2 m$ matrix and all blocks are determined by different sets of ABQ matrices.

Let $\mathrm{rank}(\mathbf{E_R}) = R$. In this case, the complexity of finding the **salt** in their attack is estimated by

$$\mathrm{Comp}_{\mathsf{FindingSalt}} = q^{l^2 m - R} \cdot (l^2 o)^3.$$

For all parameter set, we have $R = l^2 m - l + 1$. The attacker can find a set of $p = l^2 m - R$ linearly independent vectors $w_1, \ldots, w_p \in \mathbb{F}_q^{l^2 m}$ in the kernel of $\mathbf{E_R}$. Then, the attacker obtain a forgery by solving the system

$$\mathbf{w} = \mathcal{B}(\mathbf{u}_0, \mathbf{u}_0) + \sum_{i=1}^{p} y_i \mathbf{w}_i,$$

for $\mathbf{u}_0$ over $\mathbb{F}_q^{ln}$ and $y_1, \ldots, y_p \in \mathbb{F}_q$. In this case, we are working with the multi-degree $\mathbf{d}^* \in \mathbb{Z}_{\geq 0}^l$ such that

$$[\mathbf{t}^{\mathbf{d}}]H(t_1, \ldots, t_l) + p \cdot \sum_{\mathbf{d} \leq \mathbf{d}^*, \mathbf{d} \neq \mathbf{d}^*} [\mathbf{t}^{\mathbf{d}}]H(t_1, \ldots, t_l) \leq 0$$

where

$$H(t_1, \ldots, t_l) = \frac{\prod_{1 \leq i < j \leq l} (1 - t_i t_j)^{2o} \cdot \prod_{i=1}^{l} (1 - t_i^2)^o}{\prod_{i=1}^{l} (1 - t_i)^{ol - \frac{k}{l}}}.$$

The cost of this step is estimated by

$$\mathrm{Comp}_{\mathsf{Solve}} = \min_{p \leq k \leq l^2 o,\, l|k} q^{k-p} \cdot 3 \left(ol - \frac{k}{l} \right)^2 \cdot \left[\widetilde{\mathcal{M}}(\mathbf{d}^*) \right]^2 \cdot \left(2 \cdot (\log_2 q^l)^2 + \log_2 q^l \right),$$

with $\widetilde{\mathcal{M}}(\mathbf{d}^*) = \mathcal{M}(\mathbf{d}^*) + p \cdot \sum_{\mathbf{d} \leq \mathbf{d}^*, \mathbf{d} \neq \mathbf{d}^*} \mathcal{M}(\mathbf{d})$, where $\mathcal{M}(\mathbf{d})$ is the number of monomials of multi-degree exactly $\mathbf{d}$.

Therefore, the complexity of the forgery attack proposed by Cabarcas *et al* is estimated by

$$\mathrm{Comp}_{\mathsf{Forgery2}} = \max(\mathrm{Comp}_{\mathsf{FindingSalt}}, \mathrm{Comp}_{\mathsf{Solve}}). \tag{5.24}$$

It should be noted that our complexity estimation formula differs from those presented in the original paper [16]. The original paper contains minor typos in its complexity estimation formula. Initially, we were unable to reproduce the complexity results presented in their paper using the formula provided. After contacting the authors, we confirmed the corrected complexity formula through private correspondence. Using the updated formula, we were able to reproduce the complexity results reported in their paper.

Remark 5.2. We observe that the forgery attacks proposed by Beullens and Cabarcas *et al.* are significantly influenced by the rank of the $\mathbf{E_R}$ matrix. In fact, when the rank of the $\mathbf{E_R}$ matrix is sufficiently close to l^2m, those forgery attacks become ineffective. For $l = 2, 3$, the minimal rank of the $\mathbf{E_R}$ matrix is determined, and we confirmed this through exhaustive search. When $l = 4, 5$, the $\mathbf{E_R}$ matrix behaves like a random matrix. This ensures that all our parameter sets meet the security requirements defined by NIST against their forgery attacks.

In [65], we discussed matters related to the lower bound of the rank of $\mathbf{E_R}$ that ensures the Round 1 SNOVA parameter sets meet the NIST security requirements. Among these, we analyzed Beullens' forgery attack in [65]. Here, we aim to apply the same concept to two forgery attacks proposed by Beullens in Section 5.2.3 and Cabarcas *et al.* in Section 5.2.4. We calculated the corresponding lower bounds for both of these forgery attacks. Considering these two attacks, we intend to compare these lower bounds and the expected minimal rank to demonstrate that SNOVA is sufficiently secure.

With Round 2 adjustments, the result is that the matrix $\mathbf{E_R} \in \mathbb{F}_q^{ml^2 \times ml^2}$ will no longer be a block diagonal matrix with identical blocks but a $ml^2 \times ml^2$ matrix in general. The effectiveness of these forgery attacks come from the fact that the MinRank of $\mathbf{E_R}$ is not enough to resist the attacks. More precisely, if every diagonal block of $\mathbf{E_R}$ is identical, then the solution of MinRank problem of $\mathbf{E_R} = \widetilde{\mathbf{E}}_{\mathbf{R}}^{\otimes m}$ shares the same solution of MinRank problem of $\widetilde{\mathbf{E}}_{\mathbf{R}}$ which is much smaller in size. However, it is different in the case of Round 2 SNOVA, the MinRank problem will become a MinRank problem of more general matrices. This makes forgery attacks much less effective. The following table records the lower bound of the rank of $\mathbf{E_R}$ that makes SNOVA parameter set satisfy the NIST security requirements.

Table 17: The lower bounds of rank($\mathbf{E_R}$) against forgery attacks that makes SNOVA parameter sets satisfy the NIST security requirements. The numbers with "$*$" depend on the heuristic prediction.

SL	(v, o, q, l)	emr	lbd_1	$\frac{emr+lbd_1}{2}+l$	lbd_2
I	$(37, 17, 16, 2)$	67	52	61.5	60
	$(25, 8, 16, 3)$	70	49	62.5	62
	$(24, 5, 16, 4)$	77	52	68.5	67
III	$(56, 25, 16, 2)$	99	79	91	91
	$(49, 11, 16, 3)$	97	83	93	91
	$(37, 8, 16, 4)$	125	78	105.5	105
	$(24, 5, 16, 5)$	121	63	97	97^*
V	$(75, 33, 16, 2)$	131	106	120.5	122
	$(66, 15, 16, 3)$	133	110	124.5	124
	$(60, 10, 16, 4)$	157	113	139	137
	$(29, 6, 16, 5)$	146	84	120	120^*

In Table 17, "emr", "lbd_1", "lbd_2" denote the expected minimal rank of $\mathbf{E_R}$, the lower bound of rank($\mathbf{E_R}$) that ensures SNOVA parameter sets meet the NIST security requirements for the forgery attack proposed by Beullens in Section 5.2.3 and the lower bound of rank($\mathbf{E_R}$) that ensures SNOVA parameter sets meet the NIST security requirements for forgery attack proposed by Cabarcas *et al.* in Section 5.2.4, respectively.

For the parameter sets with $l = 2, 3$, we fixed the seed used to generate the ABQ matrices. In this case, the minimal rank of $\mathbf{E_R}$ is the same as the expected minimal rank. For the parameter sets with $l = 4$, the expected minimal rank of $(\mathbf{E_R})$ is emr. As mentioned above, the probability of a rank drop $d \geq 7$ is neglibile when $l \geq 4$ so using emr is justified. From the Table 17 above, it can be observed that the value $\frac{emr+lbd_1}{2} + l$ can serve as a heuristic estimate for lbd_2. For the parameter sets with $l = 5$ we estimate lbd_2 based on the heuristic estimate. As the security margin is large, more than 20 bits, the SNOVA parameter sets for $l = 5$ will satisfy the security requirements of Table 5 even if the heuristic is on the optimistic side.

5.3 Key Recovery Attacks

(v, o, q, l)-SNOVA as a (lv, lo, q)-UOV with l^2m equations. Since the SNOVA public key $[P_1], \ldots, [P_m] \in \mathrm{Mat}_{ln\times ln}(\mathbb{F}_q)$, it can be interpreted as the public key of a

(lv, lo, q)-UOV scheme. Therefore, the key recovery attacks against the UOV scheme can be executed on this (lv, lo, q)-UOV scheme. In this subsection, we analyze the structure of this $(lv, lo, q)-$UOV defined by SNOVA public key and discuss the key recovery attacks against this (lv, lo, q)-UOV. Note that the structure mentioned in this section has similar discussions in [34, 37].

For key recovery attacks against UOV and its variants, the most important task is to find the oil space $T^{-1}(\mathcal{O})$. Similarly, in SNOVA case, the task is to find the oil space of the (lv, lo, q)-UOV induced from SNOVA public key $[P_1], \ldots, [P_m]$. In conclusion, once the oil space of the related (lv, lo, q)-UOV, $T^{-1}(\mathcal{O})$ is found, then an equivalent key of SNOVA can be recovered. Here, the space $\mathcal{O}$ is defined by

$$\mathcal{O} = \{x = (x_1, \ldots, x_{ln}) \in \mathbb{F}_q^{ln} : x_1 = \cdots = x_{lv} = 0\} \tag{5.25}$$

On the other hand, since the components of $[T]$ are in $\mathbb{F}_q[S]$, the private key $[T]$ satisfying the identity over $\mathcal{R}$

$$[T] \cdot S^{\otimes n} = S^{\otimes n} \cdot [T] \tag{5.26}$$

where $S^{\otimes n} = \begin{bmatrix} S & & \\ & \ddots & \\ & & S \end{bmatrix} = S \cdot I_n$ is a $n \times n$ matrix over $\mathcal{R}$. If we identify $[T], S^{\otimes n}$ as an $ln \times ln$ matrix over $\mathbb{F}_q$ then

$$[T]^{-1}(\mathcal{O}) = [T]^{-1} S^{\otimes n}(\mathcal{O}) = S^{\otimes n} [T]^{-1}(\mathcal{O}) \tag{5.27}$$

Therefore, for each oil vector $x \in [T]^{-1}(\mathcal{O})$, we have

$$S^{\otimes n} \cdot x, \ldots, (S^{\otimes n})^{l-1} \cdot x \in T^{-1}(\mathcal{O}). \tag{5.28}$$

In particular, for any $x \in [T]^{-1}(\mathcal{O})$ and $j, k \in \{0, \ldots, l-1\}$, we then have

$$x^t \cdot (S^{\otimes n})^j [P_i] (S^{\otimes n})^k \cdot x = 0, \ (i = 1, \ldots, m). \tag{5.29}$$

Note that the Equation 5.29 directly implies that the UOV induced from the public key of SNOVA is an (lv, lo, q)-UOV scheme with $l^2 o$ equations.

(v, o, q, l)-SNOVA as (v, o, q^l)-UOVs. In [41], Nakamura *et al.* proposed a lifting method that reduces SNOVA to smaller UOV with v vinegar-variables and o oil-variables over $\mathbb{F}_{q^l}$. In [34, 37], a (v, o, q, l)-SNOVA is regarded as the a (lv, lo, q)-UOV. The components of right-upper corner T^{12} of private key $[T]$ are chosen from the subring $\mathbb{F}_q[S]$, which is generated by the symmetric matrix S.

Since the symmetric matrix S is diagonalizable over the splitting field for its characteristic polynomial. The lifting method in [41] transforms $[T]$ to a block diagonal matrix $[\widehat{T}]$ whose diagonal block components has the form of the private key for smaller (v, o, q^l)-UOVs. Namely, this is done by the Equation (9) in [41]

$$[\hat{P}_i]^{(j,j)} = [\hat{T}_j]^t \cdot [\hat{F}_i]^{(j,j)} \cdot [\hat{T}_j], \quad 1 \le j \le l, 1 \le i \le m.$$

These relations are considered as a connection between the public key and the private key of UOV with the parameter set (v, o, q^l). Specifically, by focusing on a single diagonal block of $[\widehat{T}]$, we can execute the traditional key recovery attacks on this small (v, o, q^l)-UOV.

5.3.1 Reconciliation Attack

The reconciliation attack proposed by [24] against UOV is trying to find a vector $\mathbf{o} \in \mathcal{O}$ by solving the system $P(T^{-1}(\mathbf{o})) = 0$ and hence the basis of $T^{-1}(\mathcal{O})$ can be recovered. This implies that $P(T^{-1}(\mathbf{o})) = 0$ 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 $\mathbf{o}$. 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 $\mathbb{F}_q$, is as an attack on an (lv, lo, q)-UOV which is trying to find a vector $x \in T^{-1}(\mathcal{O})$. Thus, we are in the case of solving the quadratic system

$$x^t \cdot (S^{\otimes n})^a \cdot [P_i] \cdot (S^{\otimes n})^b \cdot x = 0, \;\; (i = 1, \dots, m) \tag{5.30}$$

where $a, b \in \{0, \dots, l-1\}$, which results in l^2m equations in $lv + 1 = ln - (lo - 1)$ variables. Hence the complexity of reconciliation attack is

$$\text{Comp}_{\mathsf{Reconciliation1}} = MQ_{\mathsf{Hybrid}}(lv + 1, l^2m, q) \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \tag{5.31}$$

gates for the classical attacker.

5.3.2 Kipnis-Shamir Attack (UOV Attack)

The KS attack [36] is trying to find an equivalent private key $[T]$. If an attacker can recover $[T]$, then he can recover the oil space $T^{-1}(\mathcal{O})$. In [8, 36], it shows that $T^{-1}(\mathcal{O})$ is an invariant subspace of $[P_i']^{-1}\left[P_j'\right]$. The KS 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 [8]. A vector in $T^{-1}(\mathcal{O})$ can be expected to be found with q^{v-o} 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 KS attack remains the same.

Since the SNOVA public key $[P_1], \dots, [P_m] \in \text{Mat}_{ln \times ln}(\mathbb{F}_q)$, it can be interpreted as the public key of a (lv, lo, q)-UOV scheme. Therefore, an attacker can execute KS attack on the (lv, lo, q)-UOV induced from SNOVA public key $[P_1], \dots, [P_m]$. Thus, the complexity is

$$\text{Comp}_{\mathsf{KS}} = q^{lv-lo} \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \tag{5.32}$$

gates for the classical case.

5.3.3 Intersection Attack

In [8], 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 $P'_i(\mathbf{u_1}, \mathbf{u_2}) = \mathbf{u_1}^t [P'_i] \mathbf{u_2}$ where $[P'_i] = [P_i] + [P_i]^t$. The intersection attack is trying to first find a vector $\mathbf{y}$ in the subspace, namely the intersection $\Big([P'_i] (T^{-1}\mathcal{O}) \Big) \cap \Big([P'_j] (T^{-1}\mathcal{O}) \Big)$ where $[P'_i], [P'_j]$ are invertible, and then to obtain an equivalent key by recovering the subspace $T^{-1}(\mathcal{O})$. Since $([P'_i]^{-1})\mathbf{y}$, $([P'_j]^{-1})\mathbf{y} \in T^{-1}(\mathcal{O})$, we obtain the following system.

$$\begin{cases} P\Big(([P'_i]^{-1})\,\mathbf{y}\Big) = 0 \\ P\Big(([P'_j]^{-1})\mathbf{y}\Big) = 0 \\ P'\Big(([P'_i]^{-1})\mathbf{y}, ([P'_j]^{-1})\mathbf{y}\Big) = 0 \end{cases} \tag{5.33}$$

In case of intersection attack against SNOVA, the possible strategy is attack the (lv, lo, q)-UOV corresponding to SNOVA [34]. The attacker is trying to find a vector $\mathbf{y} \in \Big([L_1] (T^{-1}\mathcal{O}) \Big) \cap \Big([L_2] (T^{-1}\mathcal{O}) \Big)$ where $[L_1], [L_2]$ are two invertible linear combinations of the matrices $[P_i]$'s of size $ln \times ln$ over $\mathbb{F}_q$. Then, since $[L_1]^{-1}\mathbf{y}$, $[L_2]^{-1}\mathbf{y} \in T^{-1}(\mathcal{O})$, we have

$$\begin{cases} \left([L_1]^{-1}\mathbf{y}\right)^t \cdot (S^{\otimes n})^j [P_i] (S^{\otimes n})^k \cdot \left([L_1]^{-1}\mathbf{y}\right) = 0 \\ \left([L_1]^{-1}\mathbf{y}\right)^t \cdot (S^{\otimes n})^j [P_i] (S^{\otimes n})^k \cdot \left([L_2]^{-1}\mathbf{y}\right) = 0 \\ \left([L_2]^{-1}\mathbf{y}\right)^t \cdot (S^{\otimes n})^j [P_i] (S^{\otimes n})^k \cdot \left([L_1]^{-1}\mathbf{y}\right) = 0 \\ \left([L_2]^{-1}\mathbf{y}\right)^t \cdot (S^{\otimes n})^j [P_i] (S^{\otimes n})^k \cdot \left([L_2]^{-1}\mathbf{y}\right) = 0 \end{cases} \tag{5.34}$$

The case $v < 2o$. Since $\dim \Big([L_1] (T^{-1}\mathcal{O}) \Big) \cap \Big([L_2] (T^{-1}\mathcal{O}) \Big) \geq 2lo - lv > 0$, then the system 5.34 reduces to a homogeneous quadratic system of $M = 4l^2o - 2l$ equations in $N = ln - (2lo - lv - 1) = 2lv - lo + 1$ variables. Hence the complexity is

$$\mathrm{Comp}_{\mathsf{Intersection}} = MQ_{\mathsf{Hybrid}}(N, M, q) \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \tag{5.35}$$

gates.

The case $v \geq 2o$. If $n \geq 3m$, then there is no guarantee that the intersection $\Big([P'_i] (T^{-1}\mathcal{O}) \Big) \cap \Big([P'_j] (T^{-1}\mathcal{O}) \Big)$ will exist. Therefore, the intersection attack becomes a probabilistic attack against SNOVA. In this case, the complexity is

$$\mathrm{Comp}_{\mathsf{Intersection}} = \min_k q^{lv-2lo+1+k} \cdot MQ_{\mathsf{Hybrid}}(N - k + 1, M, q) \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \tag{5.36}$$

gates where $N = ln$, $M = 4l^2o - 2l$.

5.3.4 Lifting Reconciliation Attack

Following the lifting method proposed in [41], the attacker can execute the Reconciliation attack on the (v, o, q^l)-UOV scheme derived from SNOVA. Notably, for our parameter sets with $l = 2, 3, 4, 5$, the condition $mo^2 \geq vo$ is satisfied. Consequently, the (full) Reconciliation attack involves solving an overdetermined MQ system. For the lifted Reconciliation attack, our complexity estimation aligns with the estimation presented in [41].

Let $c = \min\{a \mid a^2 m > av\}$. Notably, c represents the smallest value for which the system in the Reconciliation attack becomes overdetermined. An attacker can first attempt to solve this subsystem within the (full) Reconciliation attack framework. Afterward, the remaining subsystems can be solved sequentially. Consequently, the overall complexity of the attack is dominated by the cost of solving the first subsystem.

The complexity is estimated by

$$\mathrm{Comp}_{\mathsf{LiftingReconciliation}} = MQ_{\mathsf{Hybrid}}(cv, c^2 m, q^l) \cdot (2 \cdot (\log_2 q^l)^2 + \log_2 q^l) \tag{5.37}$$

gates.

5.3.5 Lifting Kipnis-Shamir Attack

Via lifting approach in [41], a (v, o, q, l)-SNOVA can be reduced to small UOVs with parameter (v, o, q^l). Therefore, an attacker can apply KS attack to these small UOVs. The complexity is estimated by

$$\mathrm{Comp}_{\mathsf{LiftingKS}} = (q^l)^{v-o} \cdot (2 \cdot (\log_2 q^l)^2 + \log_2 q^l) \tag{5.38}$$

gates.

5.3.6 Lifting Intersection Attack

An attacker can apply the Intersection attack to this (v, o, q^l) UOV. Similar to the complexity estimation in Section 5.3.3 and [41], the complexity of lifting Intersection attack is estimated by

$$\mathrm{Comp}_{\mathsf{LiftingIntersection}} = MQ_{\mathsf{Hybrid}}(N, M, q^l) \cdot (2 \cdot (\log_2 q^l)^2 + \log_2 q^l) \tag{5.39}$$

gates where $M = 4m - 2$ and $N = n - (2o - v)$ when $v < 2o$ and

$$\mathrm{Comp}_{\mathsf{LiftingIntersection}} \tag{5.40}$$

$$= \min_k (q^l)^{v-2o+1+k+c} \cdot MQ_{\mathsf{Hybrid}}(N - k + 1, M, q) \cdot (2 \cdot (\log_2 q^l)^2 + \log_2 q^l) \tag{5.41}$$

gates where $N = \min\{n, 4m-2\}, M = 4m-2$ and $c = \max\{n-4m+2, 0\}$ when $v \geq 2o$.

5.3.7 Reconciliation Attack Proposed by Cabarcas *et al.*

In [16], the structure of the stable ideal also appears in the system of equations solved in the reconciliation attack. Therefore, the algorithm they proposed can also be applied to the reconciliation attack. In this enhanced reconciliation attack, the attacker needs to solve a SNOVA system with lv variables and l^2m equations.

In this case, we are working with the multi-degree $\mathbf{d_{sol}} \in \mathbb{Z}^l_{\geq 0}$ such that

$$[\mathbf{t}^{\mathbf{d_{sol}}}]H(t_1, \dots, t_l) \leq 0$$

where

$$H(t_1, \dots, t_l) = \frac{\prod_{1\leq i<j\leq l}(1-t_it_j)^{2o} \cdot \prod_{i=1}^{l}(1-t_i^2)^o}{\prod_{i=1}^{l}(1-t_i)^{ol-\frac{k}{l}+1}}.$$

The cost of this attack is estimated by

$$\begin{aligned} &\mathrm{Comp}_{\mathsf{Reconciliation2}} && (5.42)\\ &= \min_{1\leq k\leq l^2o,\, l|k} q^{lv-l^2o} \cdot q^k \cdot 3\left(ol - \frac{k}{l}\right)^2 \cdot \left[\overline{\mathcal{M}}(\mathbf{d_{sol}})\right]^2 \cdot (\log_2 q^l)^2 + \log_2 q^l) && (5.43)\end{aligned}$$

gates where $\overline{\mathcal{M}}(\mathbf{d_{sol}})$ is the number of monomials of multi-degree less than $\mathbf{d_{sol}}$.

In [16] and Table 13, for $l = 2, 3, 4$, we observe that the improvement proposed by Cabarcas *et al.*, compared to the Reconciliation attack, is approximately q^{l+1}.

For the parameter set with $l = 5$, computing the exact complexity of the Reconciliation 2 attack is not feasible with our current methods. However, based on the observations above, we provide a heuristic estimation for the complexity of the $l = 5$ parameter set. Specifically, we estimate the improvement brought by the Reconciliation 2 attack for the $l = 5$ parameter set as $q^{l+1} = 2^{24}$. Consequently, the heuristic complexity estimates (in $\log_2(\sharp\text{gates})$) of the Reconciliation 2 attack for the $(24, 5, 16, 5)$ parameter set is $281 - 24 = 257$, and for the $(29, 6, 16, 5)$ parameter set, it is $334 - 24 = 310$.

5.4 Side-Channel Attacks

Recently, a number of papers have appeared that study additional possibilities for side-channel attacks on SNOVA [1, 2, 46]. Masking of the Gaussian elimination step of the sign function is discussed in [46] for multiple NIST Round 2 candidates. Hardware related attacks specifically for SNOVA are studied in [2] and for more UOV signature schemes in [1]. The attacks described in these papers will be studied during Round 2. The specification and the implementation of SNOVA will be updated whenever a proposed countermeasure is considered to be either necessary or cost-effective in terms of impact on the performance.

6 Advantages and Limitations (2.B.6)

Note: This chapter is the unchanged 2.0 version. It needs to be updated.

6.1 Advantages

The main advantages of SNOVA are as follows.

- **Small public key sizes and signature sizes:** As an MQ-based signature scheme, SNOVA's signature sizes are typically small. However, SNOVA also enjoys very small public key sizes. Even at NIST security level III, we could have a pair of public key size ≈1579 bytes and signature size ≈379 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 can use two seeds combined to form a seed-type secret key, which is as small as 48 bytes.
- **A wide security margin:** We are very conservative in our security analysis, thus providing a wide security margin. To illustrate our conservatism, we note that our $l = 4$ parameter set at SL III actually satisfies the requirements at SL V for all currently known attacks.
- **Simple arithmetic:** Although the core idea of SNOVA involves the use of a noncommutative ring, the underlying basic operations to achieve the goal are essentially linear algebra. Therefore, once the nuances of the delicate design are understood, 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 be secure due to the impact of quantum computers as pointed out in [68, 69]. 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, [69] gives the corresponding condition: six times signature size and two times the public key size fit in 9KB. According to its specification, SNOVA could be a more practical general-purpose signature scheme.

6.2 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, 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 remain practical for implementation.

- **Selection of $\boldsymbol{l}$ in $\mathbf{Mat}_{l\times l}(\mathbb{F}_q)$:** 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.

References

[1] Aulbach, T., Campos, F., and Krämer, J.: **SoK: On the physical security of UOV-based signature schemes.**, Cryptology ePrint Archive, Paper 2024/1818, 2024, https://eprint.iacr.org/2024/1818.

[2] Banegas, G., Villanueva-Polanco, R.: **A Fault Analysis on SNOVA**. Cryptology ePrint Archive, Paper 2024/1883, 2024, https://eprint.iacr.org/2024/1883

[3] 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

[4] 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.

[5] 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.

[6] 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).

[7] 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).

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

[9] 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.

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

[11] Beullens, W.: **Improved Cryptanalysis of SNOVA.** Cryptology ePrint Archive, Report 2024/1297, 2024. https://eprint.iacr.org/2024/1297.pdf.

[12] Beullens, W., Campos, F., Celi, S., Hess, B., Kannwischer, M. J.: **MAYO.** https://pqmayo.org/assets/specs/mayo.pdf (version at June 1, 2023).

[13] 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)

[14] 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.

[15] 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).

[16] Cabarcas, D., Li, P., Verbel, J., Villanueva-Polanco, R.: **Improved Attacks for SNOVA by Exploiting Stability under a Group Action**. Cryptology ePrint Archive, Paper 2024/1770, 2024, https://eprint.iacr.org/2024/1770

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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

[22] 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.

[23] 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.

[24] 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).

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

[26] 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.

[27] Furue, H., Ikematsu, Y., Kiyomura, Y., Takagi, T.: **A New Variant of Unbalanced Oil and Vinegar Using Quotient Ring: QR-UOV.** In: Tibouchi, M., Wang, H. (eds) Advances in Cryptology – ASIACRYPT 2021. ASIACRYPT 2021. Lecture Notes in Computer Science(), vol 13093. Springer, Cham. https://doi.org/10.1007/978-3-030-92068-5_7.

[28] Furue, H., Ikematsu, Y.: **A New Security Analysis Against MAYO and QR-UOV Using Rectangular MinRank Attack.** In: Shikata, J., Kuzuno, H. (eds) Advances in Information and Computer Security. IWSEC 2023. Lecture Notes in Computer Science, vol 14128. Springer, Cham. https://doi.org/10.1007/978-3-031-41326-1_6

[29] 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.

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

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

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

[33] 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.

[34] Ikematsu, Y., Akiyama, R.: **Revisiting the security analysis of snova.** Cryptology ePrint Archive, Paper 2024/096, 2024. https://eprint.iacr.org/2024/096.

[35] 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.

[36] 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.

[37] Li, P., Ding, J.: **Cryptanalysis of the SNOVA signature scheme**, Cryptology ePrint Archive, Paper 2024/110, 2024, https://eprint.iacr.org/2024/110

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

[39] 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.

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

[41] Nakamura, S., Tani, Y., Furue, H.: **Lifting approach against the SNOVA scheme**. Cryptology ePrint Archive, Paper 2024/1374, 2024, https://eprint.iacr.org/2024/1374

[42] Ivica Nikolić, I., Sasaki, Y.: **A new algorithm for the unbalanced meet-in-the-middle problem.** In Jung Hee Cheon and Tsuyoshi Takagi, editors, Advances in Cryptology – ASIACRYPT 2016, pages 627–647, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

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

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

[45] 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

[46] Norga, Q., Kundu, S., Ojha, U., Ganguly, A., Karmakar, A., and Verbauwhede, I.: **Masking gaussian elimination at arbitrary order, with application to multivariate- and code-based PQC.**, Cryptology ePrint Archive, Paper 2024/1777, 2024, https://eprint.iacr.org/2024/1777.

[47] 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.

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

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] SNOVA Team: **Official SNOVA reference and AVX implementation**, 2024, GitHub repository,https://github.com/PQCLAB-SNOVA/SNOVA

[55] 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).

[56] 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).

[57] 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.

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

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

[60] 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

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

[62] 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.

[63] 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.

[64] 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.

[65] Wang, L.C., Chou, C.Y., Ding, J., Kuan, Y.L., Leegwater, J.A., Li, M.S., Tseng, B.S., Tseng, P.E., Wang, C.C.: **A Note on the SNOVA Security**. 2024. Available at https://eprint.iacr.org/2024/1517.

[66] 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

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

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

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

[70] 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.

A Rectangular SNOVA

During our process of making adjustments to SNOVA, we discovered an alternative method that ensures the $\mathbf{E_R}$ matrix achieves full rank. As a byproduct of our research, we provide a brief introduction to this variant in the Appendix. Through this approach, we aim to demonstrate the flexibility of the SNOVA scheme and its variants.

Let v, o be positive integers with $v > o$ and $\mathbb{F}_q$ be the finite field of order q. Let $n = v + o$ and $m = o$. Let r be a positive integer. The public map of a (v, o, q, l, r) Rectangular SNOVA variant scheme is defined as follows:

We construct the public key $[P_1], \ldots, [P_m]$ via the congruence relation

$$[P_1] = \left[P_{1,j,k}\right] = [T]^t [F_1] [T], \ldots, [P_m] = \left[P_{m,j,k}\right] = [T]^t [F_m] [T].$$

Furthermore, we construct additional r public key $[P_{m+1}], \ldots, [P_{m+r}]$.

The public map of SNOVA is $\widetilde{P} = \widetilde{F} \circ T$. For $i \in \{1, 2, \ldots, m\}$, $\widetilde{P_i} = \widetilde{F_i} \circ T$. We define $\widetilde{P_i}$ as

$$\widetilde{P_i}(\mathbf{U}) = \widetilde{F_i}(T(\mathbf{U})) = \sum_{\alpha=0}^{l^2+l-1} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{i,\alpha} \cdot U_j^t (Q_{i,\alpha,1} P_{i',jk} Q_{i,\alpha,2}) U_k \cdot B_{i,\alpha}$$

where $i' = (i + \alpha) \mod (m + r)$ and $P_{i',jk}$ is the (j,k)-th entry of matrix $[P_{i'}]$.

Public key. The public key consists of the matrices $[P_i]$ and the matrices $A_{i,\alpha}$, $B_{i,\alpha}$, $Q_{i,\alpha,1}$ and $Q_{i,\alpha,2}$ for $i = 1, \ldots, m + r$ and for $\alpha = 0, 1, \ldots, l^2 + l - 1$, or simply the seed $\mathbf{s_{public}}$ that generates them. By utilizing matrices $[P_i]$ and the seed $\mathbf{s_{public}}$, the verifier is able to obtain the public map $\widetilde{P}$ and subsequently verify the received signature.

Remark A.1. Compared to SNOVA, the introduction of additional r public keys slightly increases the size of the public key. At the same time, the corresponding $\mathbf{E_R}$ matrix becomes a $m \times (m + r)$ rectangular matrix. In this scenario, $\mathbf{E_R}$ can easily achieve full rank.

Remark A.2. The rectangular public map introduced above can also be written as

$$\widetilde{P_i}(\mathbf{U}) = \widetilde{F_i}(T(\mathbf{U})) = \sum_{\alpha=0}^{l^2+l-1} \sum_{i'=1}^{m+r} C_{i,i'}^{(\alpha)} \sum_{j=1}^{n} \sum_{k=1}^{n} A_{i,\alpha} \cdot U_j^t (Q_{i,\alpha,1} P_{i',jk} Q_{i,\alpha,2}) U_k \cdot B_{i,\alpha}$$

where $C_{i,i'}^{(\alpha)} = 1$ if $i' = (i + \alpha) \mod (m + r)$ and $C_{i,i'}^{(\alpha)} = 0$ for all other values. Another extension of SNOVA is to use another choice for $C_{i,i'}^{(\alpha)}$ such as a set of random matrices.