IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 1, pp. 245–269. DOI:10.46586/tches.v2022.i1.245-269

Efficient Implementations of Rainbow and UOV using AVX2

Kyung-Ah Shim[1†], Sangyub Lee[1], Namhun Koo[2]

[1] National Institute for Mathematical Sciences, Daejeon, Republic of Korea,
{kashim,sylee}@nims.re.kr
[2] Ewha Womans University, Seoul, Republic of Korea,
nhkoo@ewha.ac.kr

Abstract. A signature scheme based on multivariate quadratic equations, Rainbow, was selected as one of digital signature finalists for NIST Post-Quantum Cryptography Standardization Round 3. In this paper, we provide efficient implementations of Rainbow and UOV using the AVX2 instruction set. These efficient implementations include several optimizations for signing to accelerate solving linear systems and the Vinegar value substitution. We propose a new block matrix inversion (BMI) method using the Lower-Diagonal-Upper decomposition of blocks matrices based on the Schur complement that accelerates solving linear systems. Compared to UOV implemented with Gaussian elimination, our implementations with the BMI result in speedups of 12.36%, 24.3%, and 34% for signing at security categories I, III, and V, respectively. Compared to Rainbow implemented with Gaussian elimination, our implementations with the BMI result in speedups of 16.13% and 20.73% at the security categories III and V, respectively. We show that precomputation for the Vinegar value substitution and solving linear systems dramatically improve their signing. UOV with precomputation is 16.9 times, 35.5 times, and 62.8 times faster than UOV without precomputation at the three security categories, respectively. Rainbow with precomputation is 2.1 times, 2.2 times, and 2.8 times faster than Rainbow without precomputation at the three security categories, respectively. We then investigate resilience against leakage or reuse of the precomputed values in UOV and Rainbow to use the precomputation securely: leakage or reuse of the precomputed values leads to their full secret key recoveries in polynomial-time.

Keywords: Block Matrix Inversion · Digital Signature · Gaussian Elimination · Multivariate-Quadratic Problem · Post-Quantum Cryptography · Precomputation · Schur Complement

1 Introduction

Developments of a quantum computer have inspired great interest in post-quantum cryptographic primitives that are believed to remain secure against a quantum computer. Cryptographic primitives based on multivariate quadratic (MQ) equations are one of promising post-quantum replacements for current public-key cryptographic algorithms based on the discrete logarithm problem and the integer factorization problem. In 2020, NIST selected seven finalists for Post-Quantum Cryptography (PQC) Standardization Round 3 [ABC+20, SAB+20, CDH+20, DKR+20, LDK+20, PFH+20, DCP+20b]. The third-round finalists for digital signatures are two lattice-based schemes, Dilithium and

†Corresponding Author. This research was supported by the National Institute for Mathematical Sciences funded by Ministry of Science and ICT of Korea (B21720000).

Received: 2021-07-15 Accepted: 2021-09-15 Published: 2021-11-19

Falcon, and an MQ-scheme, Rainbow. These finalists will be considered for standardization at the end of the third round.

MQ-schemes mainly rely on the hardness of solving large systems of multivariate quadratic equations, called the MQ-problem. To hide a trapdoor, they require an Affine-Substitution-Affine (ASA) structure related to the Extended Isomorphism of Polynomials (EIP) problem [Pat96]. Since Imai and Matsumoto [MI88] introduced the first MQ-encryption scheme, most of the MQ-schemes have been broken except few signature schemes including Unbalanced Oil-and-Vinegar (UOV) variants [KPG99, DS05] and HFEv-variants [PCG01, PCY+15]. MQ-schemes require only modest computational resources and are suitable for resource constrained devices [BERW08, CCC+09]: it is easy to implement requiring simple operations such as matrix-vector products and solving linear systems over small finite fields without multi-precision arithmetic. Thus, they can be efficiently implemented on low cost devices, without the need of a cryptographic coprocessor. Their signature length and performance are the shortest and fastest among known post-quantum signature schemes, respectively. In spite of fast performance and short signatures, they suffer from relatively large key sizes. However, in medium devices with sufficient memory capabilities such as smartphones, large key size is not a major problem.

UOV and Rainbow are based on two hard problems, the MQ-problem and the EIP-problem, while Rainbow additionally requires the hardness of the MinRank problem due to its multi-layered structure. New methods to improve the MinRank attack are still being developed [PS20, VBC+19, BBB+20, BBC+20]. Rainbow team [DCP+20a] modified their parameters based on these new security analyses by increasing the number of equations and variables. Ward Beullens [Beu20] gave new intersection attacks on UOV and Rainbow, and MinRank attacks on Rainbow. He claimed that the attacks reduced the cost of a key recovery by a factor of $2^{16}, 2^{30}$, and 2^{46} for the parameter sets of Rainbow submitted to NIST PQC at three security categories, respectively. Subsequently, Rainbow team showed that the Rainbow parameters submitted to Round 3 still meet the NIST level I, III, and V security requirements in their updated security model and analysis without detracting from Ward's work [DCP+20c].

Main Results. A main idea to invert a system of quadratic equations in UOV and Rainbow is to convert the quadratic system to a linear system by substituting random Vinegar values into the Vinegar variables of the central quadratic equations. Signing of UOV and Rainbow is dominated by these two operations: the Vinegar value substitution and solving linear systems. In this paper, we present optimizations to accelerate the two major operations via a block matrix inversion method and precomputation.

- **Implementations of UOV/Rainbow.** We select new parameters of UOV secure against the recent attacks including known algebraic attacks at the three security categories. We then implement UOV with the new parameters and Rainbow with the modified parameters submitted to NIST PQC at the three security categories I, III, and V [DCP+20a]. After profiling their implementations, we determine target optimizations.

- **Efficient Implementations of UOV/Rainbow.** We propose two optimizations to improve signing of UOV and Rainbow, and present their efficient implementations in C program language on an Intel 64-bit processor using the AVX2 instruction set.

  - **Block Matrix Inversion (BMI) Method.** We propose a BMI method using the Lower-Diagonal-Upper decomposition of block matrices based on the Schur complement. We use the BMI method to reduce the size of the matrix being inverted by half for accelerating solving linear systems. Compared to UOV implemented with Gaussian elimination, we speed up the signing by 12.36%, 24.3%, and 34% at the security categories I, III, and V, respectively. Compared

to Rainbow implemented with Gaussian elimination, we speed up the signing by 16.13% and 20.73% at the security categories III and V, respectively.

- **Precomputation.** We present implementations of UOV and Rainbow with precomputation for message independent operations that dramatically improve signing. UOV with precomputation is 16.9 times, 35.5 times, and 62.8 times faster than UOV without precomputation at the three security categories, respectively, while Rainbow with precomputation is 2.1 times, 2.2 times, and 2.8 times faster than Rainbow without precomputation at the three security categories, respectively. Rainbow is faster than UOV, while UOV with precomputation is about 2.7 times, 7.3 times, and 12.2 times faster than Rainbow with precomputation at the three security categories, respectively.
- **Resilience against Leakage or Reuse of the Precomputed Values.** We investigate resilience of leakage or reuse of the precomputed values in UOV and Rainbow to use the precomputation values securely. We show that if $(n+1)$ signatures generated by the revealed precomputed values are given then the secret keys of UOV and Rainbow are completely recovered in polynomial-time. If $(m+1)$ and (o_2+1) signatures generated by reusing the precomputed values are given then the secret keys of UOV and Rainbow are entirely recovered in polynomial-time, respectively.

Organization. Section 2 describes Rainbow and UOV specifications. In Section 3, we select new parameters of UOV secure against the recent attacks and we determine target optimizations after profiling their implementations. In Section 3, we present the BMI method for solving linear systems and investigate the actual improvements in terms of matrix sizes and the security levels. In Section 4, we propose UOV and Rainbow with precomputation and then investigate resilience against leakage or reuse of the precomputed values. In each section, we present efficient UOV/Rainbow implementations with our optimizations using the AVX2 instruction set and investigate their speedups compared to the original schemes. We conclude this paper in Section 5.

2 Preliminaries

We describe key generation, signing and verification algorithms of Rainbow and UOV.

2.1 Rainbow and UOV

Main Parameters.

- $\mathbb{F}_q$: the finite field of q elements
- m: the number of polynomials in the public key
- v: the number of Vinegar variables
- o: the number of Oil variables in UOV, $m = o$
- o_i: the number of Oil variables in the i-th layer of Rainbow, $m = o_1 + o_2$
- n: the number of variables in the public key, $n = m + v$.

We describe Rainbow with two layers. Let v, o_1, and o_2 be positive integers such that $m = o_1 + o_2$ and $n = v + m$. Define sets of integers

$$V = \{1, \cdots, v\},\ O_1 = \{v+1, \cdots, v+o_1\},\ O_2 = \{v+o_1+1, \cdots, v+o_1+o_2\}.$$

A central map $\mathcal{F}(\mathbf{x}) = (\mathcal{F}^{(1)}(\mathbf{x}), \cdots, \mathcal{F}^{(m)}(\mathbf{x}))$ is a system of m multivariate quadratic polynomials in n variables $x_1, \cdots, x_n$ defined by

$$\mathcal{F}^{(i)}(\mathbf{x}) = \sum_{s \in O_1, t \in V} \alpha_{st}^{(i)} x_s x_t + \sum_{s,t \in V, s \leq t} \beta_{st}^{(i)} x_s x_t + \sum_{s \in V \cup O_1} \gamma_s^{(i)} x_t + \eta^{(i)},\ 1 \leq i \leq o_1$$

$$\mathcal{F}^{(i)}(\mathbf{x}) = \sum_{s \in O_2, t \in V \cup O_1} \alpha_{st}^{(i)} x_s x_t + \sum_{s,t \in V \cup O_1, s \leq t} \beta_{st}^{(i)} x_s x_t + \sum_{s \in V \cup O_1 \cup O_2} \gamma_s^{(i)} x_t + \eta^{(i)},\ o_1 + 1 \leq i \leq m$$

Two invertible affine maps $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ are needed to destroy the special structure of $\mathcal{F}$. A public key is given by $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ and a secret key is $(\mathcal{S}, \mathcal{F}, \mathcal{T})$.

Rainbow

- **KeyGen**(1^λ)**.** For a security parameter λ, output a public key as $PK = \mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ and a secret key as $SK = (\mathcal{S}, \mathcal{F}, \mathcal{T})$.
- **Sign**$(SK, \lambda, \mathtt{m})$**.** Given a message $\mathtt{m}$ and a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}_q^m$,
  - Choose a λ-bit random salt r, compute $\mathbf{h} = \mathcal{H}(\mathtt{m}, r) \in \mathbb{F}_q^m$ and $\mathbf{a} = \mathcal{S}^{-1}(\mathbf{h})$.
  - Compute $\mathbf{b} = \mathcal{F}^{-1}(\mathbf{a})$, i.e. $\mathcal{F}(\mathbf{b}) = \mathbf{a}$ as follows:
    - In the first layer, select Vinegar values $(s_1, \cdots, s_v) \in \mathbb{F}_q^v$ at random and obtain a linear system of o_1 equations with o_1 unknowns $x_{v+1}, \cdots, x_{v+o_1}$ by substituting $(s_1, \cdots, s_v)$ into o_1 central polynomials $\mathcal{F}^{(k)}$ for $1 \leq k \leq o_1$. After that, find a solution $(s_{v+1}, \cdots, s_{v+o_1})$ of the linear system using Gaussian elimination.
    - In the second layer, obtain a linear system of o_2 equations with o_2 unknowns $x_{v+o_1+1}, \cdots, x_n$ by substituting $(s_1, \cdots, s_{v+o_1})$ into o_2 central polynomials $\mathcal{F}^{(i)}$ for $o_1 + 1 \leq i \leq m$. After that, find a solution $(s_{v+o_1+1}, \cdots, s_n)$ of the linear system using Gaussian elimination. Then $\mathbf{b} = (s_1, \cdots, s_n)$.
    - If one of the linear systems is not solvable, choose another random number r' and try again.
  - Compute $\sigma = \mathcal{T}^{-1}(\mathbf{b})$ and output $\tau = (\sigma, r)$ as a signature on $\mathtt{m}$.
- **Verify**$(PK, \mathtt{m}, \tau)$**.** Given a signature $(\tau, \mathtt{m})$ and the public key $\mathcal{P}$, check the equality $\mathcal{P}(\sigma) = \mathcal{H}(\mathtt{m}, r)$. If the equality holds, output *valid.*

UOV

- **KeyGen**(1^λ)**.** For a security parameter λ, a public key is $PK = \mathcal{P} = \mathcal{F} \circ \mathcal{T}$ and a secret key is $SK = (\mathcal{F}, \mathcal{T})$. UOV does not require an affine map S since all the central polynomials have the same form.
- **Sign**$(SK, \mathtt{m})$**.** It is the same as that of Rainbow's first layer.
- **Verify**$(PK, \mathtt{m}, \sigma)$**.** It is the same as that of Rainbow.

Algorithm 1, Algorithm 2, and Algorithm 3 specify key generation, signing, and verification of Rainbow submitted to NIST PQC Standardization Round 3 [DCP+20b]. The description of UOV is consistent with a single layer version of Rainbow. Algorithm 4, Algorithm 5, and Algorithm 6 specify key generation, signing, and verification of UOV. Their signing and verification algorithms use the hash value $\mathcal{H}(\mathcal{H}(\mathtt{m})||r)$ instead of $\mathcal{H}(\mathtt{m}, r)$ for efficiency.

Algorithm 1 Rainbow Key Generation

Require: Rainbow parameters (q, v_1, o_1, o_2), length of salt l.
Ensure: Rainbow key pair (sk, pk).

$m \leftarrow o_1 + o_2$
$n \leftarrow m + v_1$
repeat
  $M_S \leftarrow \texttt{Matrix}(q, m, m)$
until IsInvertible(M_S) == **TRUE**
$\mathcal{S} \leftarrow M_S$
$Inv\mathcal{S} \leftarrow M_S^{-1}$
repeat
  $M_T \leftarrow \texttt{Matrix}(q, n, n)$
until IsInvertible(M_T) == **TRUE**
$\mathcal{T} \leftarrow M_T$
$Inv\mathcal{T} \leftarrow M_T^{-1}$
$\mathcal{F} \leftarrow \texttt{Rainbowmap}(q, v_1, o_1, o_2)$
$\mathcal{P} \leftarrow \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$
$sk \leftarrow (Inv\mathcal{S}, \mathcal{F}, Inv\mathcal{T}, l)$
$pk \leftarrow (\mathcal{P}, l)$
Return (sk, pk)

Algorithm 2 Rainbow Signature Generation

Require: document d, Rainbow private key ($InvS$, $\mathcal{F}$, $InvT$), length of the salt l.
Ensure: signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$ such that $\mathcal{P}(\mathbf{z}) = \mathcal{H}(\mathcal{H}(d)||r)$.

repeat
  $y_1, ..., y_{v_1} \leftarrow_R \mathbb{F}_q$
  $\hat{f}^{(v_1+1)}, ..., \hat{f^{(n)}} \leftarrow f^{(v_1+1)}(y_1, ..., y_{v_1}), ..., f^{(n)}(y_1, ..., y_{v_1})$
  $(\hat{F}, C_F) \leftarrow \texttt{Aff}^{-1}(\hat{f}^{(v_1+1)}, ..., \hat{f}^{(n)})$
until IsInvertible$(\hat{F})$ == **TRUE**
$InvF = \hat{F}^{-1}$
repeat
  $r \leftarrow \{0,1\}^l$
  $\mathbf{h} \leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
  $\mathbf{x} \leftarrow InvS \cdot \mathbf{h}$
  $(y_{v_1+1}, ..., y_{v_2}) \leftarrow InvF \cdot ((x_{v_1+1}, ..., x_{v_2}) - C_F)$
  $\hat{f}^{(v_2+1)}, ..., \hat{f}^{(n)} \leftarrow \hat{f}^{v_2+1}(y_{v_1+1}, ..., y_{v2}), ..., \hat{f}^{(n)}(y_{v_1+1}, ..., y_{v2})$
  $t, (y_{v_2+1}, ..., y_n) \leftarrow Gauss(\hat{f}^{(v_2+1)} = x_{v_2+1}, ..., \hat{f}^{(n)} = x_n)$
until t == **TRUE**
$\mathbf{z} = InvT \cdot \mathbf{y}$
$\sigma \leftarrow (\mathbf{z}, r)$
Return σ

Algorithm 3 Rainbow Signature Verification

Require: document d, signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$.
Ensure: boolean value **TRUE** or **FALSE**.

h $\leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
h' $\leftarrow \mathcal{P}(\mathbf{z})$
if h' == h then
  return TRUE
else
  return FALSE
end if

Algorithm 4 UOV Key Generation

Require: UOV parameters (q, v, o), length of salt l.
Ensure: Rainbow key pair (sk, pk).

$m \leftarrow o$
$n \leftarrow m + v$
repeat
    $M_T \leftarrow \texttt{Matrix}(q, n, n)$
until IsInvertible(M_T) == **TRUE**
$\mathcal{T} \leftarrow M_T$
$Inv\mathcal{T} \leftarrow M_T^{-1}$
$\mathcal{F} \leftarrow \texttt{UOVmap}(q, v, o)$
$\mathcal{P} \leftarrow \mathcal{F} \circ \mathcal{T}$
$sk \leftarrow (\mathcal{F}, Inv\mathcal{T}, l)$
$pk \leftarrow (\mathcal{P}, l)$
Return (sk, pk)

Algorithm 5 UOV Signature Generation

Require: document d, UOV private key ($\mathcal{F}$, $InvT$), length of the salt l.
Ensure: signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$ such that $\mathcal{P}(\mathbf{z}) = \mathcal{H}(\mathcal{H}(d)||r)$.

repeat
    $y_1, ..., y_v \leftarrow_R \mathbb{F}_q$
    $\hat{f}^{(v_1+1)}, ..., \hat{f^{(n)}} \leftarrow f^{(v+1)}(y_1, ..., y_v), ..., f^{(n)}(y, ..., y_v)$
    $(\hat{F}, C_F) \leftarrow \texttt{Aff}^{-1}(\hat{f}^{(v+1)}, ..., \hat{f}^{(n)})$
until IsInvertible$(\hat{F})$ == **TRUE**
$InvF = \hat{F}^{-1}$
$r \leftarrow \{0,1\}^l$
$\mathbf{x} \leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
$(y_{v+1}, ..., y_n) \leftarrow InvF \cdot ((x_{v+1}, ..., x_n) - C_F)$
$\mathbf{z} = InvT \cdot \mathbf{y}$
$\sigma \leftarrow (\mathbf{z}, r)$
Return σ

Algorithm 6 UOV Signature Verification

Require: document d, signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$.
Ensure: boolean value **TRUE** or **FALSE**.

h $\leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
h' $\leftarrow \mathcal{P}(\mathbf{z})$
if h' == h then
    return TRUE
else
    return FALSE
end if

3 Implementations of UOV and Rainbow

We first consider major computations in UOV/Rainbow to improve their signing. We then select new secure parameters of UOV against the recent attacks including known algebraic attacks and implement them using the AVX2 instruction set. To identify how much improvement needs to be made in their implementations, we profile the implementations and determine optimization strategies.

3.1 Major Computations of UOV and Rainbow

We consider major computations in UOV and Rainbow.

Major Computations in UOV and Rainbow. A main idea to invert a system of quadratic equations in UOV and Rainbow is to convert the quadratic system to a linear system by substituting random Vinegar values into the Vinegar variables of the central quadratic polynomials. There are two types of computations in signing:

- **Substitution of Vinegar Values into the Central Polynomials.** Calculations for substituting random Vinegar values into the central polynomials are required. Since there are a large number of quadratic terms with Vinegar×Vinegar indexes and Vinegar×Oil indexes being substituted by the Vinegar values, the computations are heavy.
- **Solving Linear System.** Solving the linear systems after the Vinegar value substitution are required. Gaussian elimination is used to find a solution of the linear system, whose complexity is $O(k^3)$ for a $k \times k$ random matrix.

Thus, signing requires $O(k^3)$ complexity, where k is the number of equations in the linear system. These computations are the main bottlenecks for signing cost.

Parameters of UOV and Rainbow are given by $(\mathbb{F}_q, v, o)$, where $m = o$ and $n = v + m$, and $(\mathbb{F}_q, v, o_1, o_2)$, where $m = o_1 + o_2$ and $n = v + m$, respectively. The parameters related to the computational burden are as follows:

- **Large Numbers of Variables (n).** UOV and Rainbow require the large numbers of variables which cause heavy computational cost for the Vinegar value substitution.
- **Large Sizes of Matrices being Inverted (m, o_2).** The numbers of equations (m) in UOV and Rainbow are determined by the complexities against direct attacks. To solve the linear systems,
  - UOV with a single layer requires one inversion of an $m \times m$ matrix, where $m(= o)$ is at most twice o_i $(i = 1, 2)$, where o_1 and o_2 are the numbers of equations in the first and second layers of Rainbow, respectively.
  - Rainbow with two layers requires two inversions of an $o_1 \times o_1$ matrix and an $o_2 \times o_2$ matrix, where $m = o_1 + o_2$, and o_i is smaller than o in UOV. In particular, according to the security analysis of Rainbow [BBC+20, DCP+20a], a key formula for complexity includes the number k, which corresponds to o_2. The modified Rainbow parameters submitted to NIST PQC were chosen $o_2 > o_1$ [DCP+20a]. Since the increase of k will increase the attack complexity, o_2 is getting bigger.

3.2 Parameter Selection and Implementations of UOV and Rainbow

Our implementations are based on the open source codes of Rainbow submitted to NIST PQC Standardization Round 3 [DCP+20b]. After implementing the schemes, we profile their implementations.

Target Platform. The computer we have used is equipped with an Intel(R) Core(TM) i9-10900X CPU running at the constant clock frequency of 3.70GHz.

Random Number Generation and Hashing. We use AES_CTR_DRBG as the random number generator. We use SHA-2 as the underlying hash function. In the SHA-2

Table 1: Suggested parameters of UOV/Rainbow at three security categories.

Scheme	Security Category	I	III	V
UOV	λ (Gates)	2^{146}	2^{212}	2^{274}
	$(\mathbb{F}_q, o, v)$	$(\mathbb{F}_{2^8}, 46, 70)$	$(\mathbb{F}_{2^8}, 72, 109)$	$(\mathbb{F}_{2^8}, 96, 144)$
	Direct Attack	$2^{146.05}$	$2^{212.05}$	$2^{274.847}$
	Intersection Attack	$2^{166.87}$	$2^{236.36}$	$2^{291.501}$
Rainbow	λ (Gates)	2^{147}	2^{217}	2^{281}
	$(\mathbb{F}_q, v, o_1, o_2)$	$(\mathbb{F}_{2^4}, 36, 32, 32)$	$(\mathbb{F}_{2^8}, 68, 32, 48)$	$(\mathbb{F}_{2^8}, 96, 36, 64)$
	Direct Attack	2^{164}	2^{234}	2^{285}
	MinRank Attack	2^{162}	2^{228}	2^{296}
	RBS Attack	2^{147}	2^{217}	2^{281}

hash function family, we use SHA256, SHA384, and SHA512 with output lengths of 256, 384, and 512 bits, respectively.

Selection of Finite Fields. We choose $\mathbb{F}_q = \mathbb{F}_{2^8}$ for UOV and $\mathbb{F}_q = \mathbb{F}_{2^8}$ or $\mathbb{F}_{2^4}$ for Rainbow as the underlying finite fields.

The Use of Random Salts. We use a random salts $r \in \{0,1\}^l$ to achieve provable security as in [SSH11] which should be used only once.

Use of Equivalent Key and Linear Maps. UOV implementation uses a secret key $\mathcal{T}$ as an equivalent key of UOV of the form $\mathcal{T} = \begin{pmatrix} I & T' \\ 0 & I \end{pmatrix}$ and a linear map. Rainbow implementation uses a secret key $(\mathcal{S}, \mathcal{T})$ as an equivalent key of Rainbow of the form $\mathcal{S}^{-1} = \begin{pmatrix} I & S' \\ 0 & I \end{pmatrix}$, $\mathcal{T} = \begin{pmatrix} I & T^{(1)} & T^{(2)} \\ 0 & I & T^{(3)} \\ 0 & 0 & I \end{pmatrix}$ and linear maps.

Selection of Secure Parameters. For implementations of UOV and Rainbow, we need to select their secure parameters.

- For UOV, we need to select new secure parameters of UOV against the attacks in [Beu20] including known algebraic attacks. In [CHT12], $\mathrm{UOV}(\mathbb{F}_q, o, v) = (\mathbb{F}_{2^8}, 44, 59)$ was suggested as a secure parameter at a 128-bit security level. The intersection attack on $\mathrm{UOV}(\mathbb{F}_{2^8}, 44, 59)$ [Beu20] requires only 2^{95} multiplications to recover the secret key, which is much less than the claimed security 2^{128}. We suggest new secure parameters of UOV at the three security categories summarized in Table 1, where λ is the required number of gates for each security category [DCP+20b].
- For Rainbow, we use the modified parameters submitted to NIST at the three security categories [DCP+20a]. The parameters at the three security categories are summarized in Table 1. Complexities in Table 1 are the lowest complexities that determine the corresponding security levels. RBS attack stands for the Rainbow Band Separation attack.

UOV/Rainbow Implementation Results. We provide implementations of UOV and Rainbow based on codes submitted to NIST PQC Standardization Round 3 [DCP+20b] using the AVX2 instruction set on the target platform.

Table 2: UOV/Rainbow implementation results using AVX2 at three security categories in CPU cycles.

Scheme	Security Category	I	III	V
UOV	KeyGen.	29 077 126	98 870 925	161 016 435
	Sign	201 834	707 959	1 486 775
	Verify	125 312	222 012	485 344
Rainbow	KeyGen.	11 575 144	65 099 975	214 977 689
	Sign	68 203	322 799	807 309
	Verify	46 857	151 466	395 259

- The results presented in Table 2 include the numbers of CPU cycles required by the key generation, signing, and verification.
- Each result is an average of 10,000 measurements for each function using the C programming language with GNU GCC version 10.1.0 compiler on Centos 7.9.2009. Hyperthreading and Turbo Boost are switched off.

Profiling UOV/Rainbow Implementation. To identify how much improvement needs to be made in the implementations, we profile their signing given in Table 3.

- In Table 3, Vinegar value substitution represents the process of substituting Vinegar values into the central polynomials, $\mathcal{F}^{(i)}$ for $i = 1, \cdots, m$ in UOV ($i = 1, \cdots, o_1$ and $i = o_1 + 1, \cdots, m$ in the first and second layers of Rainbow, respectively) after choosing random Vinegar values $s_V = (s_1, \cdots, s_V) \in \mathbb{F}_q^v$. An $m \times m$ matrix, LS_V, is the coefficient matrix of the linear system obtained from the Vinegar value substitution in UOV. Two $o_1 \times o_1$ and $o_2 \times o_2$ matrices, $LS_{V,1}$ and $LS_{V,2}$, are the coefficient matrices of the linear systems obtained from the Vinegar value substitution in the first and second layers of Rainbow, respectively. Etc. represents the rest of operations except the above two operations.
- Run-time of signing in UOV and Rainbow is mostly dominated by the two operations. In UOV, the proportions of cycles spent in the Vinegar values substitution and in finding an inverse matrix are up to 58% and 48%, respectively. In Rainbow, the proportions of cycles spent in the two major operations are up to 63% and 42%, respectively.
- Thus, optimizing these two operations are going to provide a large speedup.
  - To accelerate solving linear systems, we will reduce the sizes of matrices being inverted by half based on a block matrix inversion method.
  - We will use precomputation to handle both of the Vinegar value substitution and solving linear systems.

Timing Attack Protection. As in the implementation of Rainbow [DCP+19], all key dependent operations of our implementations of UOV/Rainbow are performed in a time-constant manner.

4 Efficient Implementations of UOV and Rainbow

In this section, we propose a block matrix inversion method and precomputation for solving linear systems and the Vinegar value substitution to accelerate signing of UOV

Table 3: Profiling UOV/Rainbow implementations at three security categories.

Scheme	Layer	Operations	I	III	V
UOV	1	Vinegar value substitution	58.09%	48.37%	56.60%
		Computation of LS_V^{-1}	36.38%	47.98%	41.71%
		Etc.	5.53%	3.65%	1.69%
Rainbow	1	Vinegar value substitution	19.15%	18.58%	34.37%
		Computation of $LS_{V,1}^{-1}$	27.04%	11.37%	8.03%
		Etc.	3.49%	1.26%	0.68%
	2	Vinegar value substitution	24.08%	39.54%	29.07%
		Computation of $LS_{V,2}^{-1}$	15.00%	25.38%	25.34%
		Etc.	11.24%	3.87%	2.51%

and Rainbow. For our implementations, we utilize Rainbow team's AVX2-specialized basic functions for GF arithmetic, basic linear algebra, and matrix operations, and constant-time basic functions in its source code submitted to NIST PQC Round 3. For the BMI implementation, we use its AVX2 specialized functions for matrix operations and (constant-time) Gaussian elimination. Implementations for precomputation are constructed with rearranging the computational operations and adjusting the memory operations.

4.1 A Fast Method for Solving Linear Systems: Block Matrix Inversion

Signing of UOV and Rainbow uses Gaussian elimination to get LS_V^{-1}. The size of a matrix being inverted is one of the reasons for heavy computation which requires $O(m^3)$ complexity for inverting an $m \times m$ matrix. Now, we propose a new method, block matrix inversion method, to reduce the size of a matrix being inverted for accelerating UOV/Rainbow signing. We then investigate the actual improvements of our method in their implementations at various security levels.

Reduce the Size of a Matrix being Inverted. We first consider a nonsingular square matrix R and its inverse R^{-1} that can be partitioned into 2×2 blocks as

$$R = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \text{ and } R^{-1} = \begin{pmatrix} E & F \\ G & H \end{pmatrix}.$$

The matrices R and R^{-1} must have even dimension in the all-square partitions. We consider the square diagonal partition of R and R^{-1}. In this case, A, D, E, H are square matrices, A and E have the same size, and so do D and H. We use the following well-known theorem [ND+77].

Theorem 1. Let R be a matrix partitioned into 2×2 blocks.

(i) Assume A is nonsingular: then the matrix R is invertible if and only if the Schur complement $(D - CA^{-1}B)$ of A is invertible and

$$R^{-1} = \begin{pmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{pmatrix}.$$

(ii) Assume D is nonsingular: then the matrix R is invertible if and only if the Schur complement $(A - BD^{-1}C)$ is invertible and

$$R^{-1} = \begin{pmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -D^{-1}C(A - BD^{-1}C)^{-1} & D^{-1} + D^{-1}C(A - BD^{-1}C)^{-1}BD^{-1} \end{pmatrix}.$$

The inversion method in Theorem 1 requires two inversions for A^{-1} and $[D-CA^{-1}B]^{-1}$, six matrix multiplications of the half-sized block matrices and three additive operations. To reduce the required operations, we use the Lower-Diagonal-Upper (LDU) decomposition of block matrices based on the Schur complement. We give an efficient block matrix inversion method for computing $R^{-1} \cdot \alpha$ to reduce the operations by more than half in Theorem 2.

Theorem 2. For a nonsingular $k \times k$ matrix R in the above, $R^{-1} \cdot \alpha$ requires two inversions, two matrix multiplications of the half-sized block matrices and four block matrix-vector products, where k is even and $\alpha = (\alpha_1, \cdots, \alpha_{k/2})^{\mathtt{T}}$.

Proof. A nonsingular square matrix R of 2×2 blocks is represented by the LDU decomposition of block matrices based on the Schur complement as

$$R = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & O \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & O \\ 0 & D - CA^{-1}B \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix} = L \cdot D_{Sc} \cdot U.$$

Thus, R^{-1} can be expressed by A^{-1} and the inverse of the Schur complement of A, $[D - CA^{-1}B]^{-1}$, if they exist,

$$\begin{aligned} R^{-1} &= U^{-1} \cdot D_{Sc}^{-1} \cdot L^{-1} \\ &= \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & O \\ 0 & [D - CA^{-1}B]^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix}. \end{aligned}$$

For computing $R^{-1} \cdot \alpha = R^{-1} \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix}$, we reduce the matrix multiplications of the method in Theorem 1 as follows: After computing $A^{-1}B$ and $C(A^{-1}B)$, calculate A^{-1} and $[D - CA^{-1}B]^{-1}$, where two matrix multiplications and two inversions of the block matrices. All remaining computations are made by four block matrix-vector products. We get

$$R^{-1} \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix} = \begin{pmatrix} I & -A^{-1}B \\ 0 & I \end{pmatrix} \begin{pmatrix} A^{-1} & O \\ 0 & [D - CA^{-1}B]^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix}.$$

We obtain $(CA^{-1}) \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix}$ by computing $C \left[A^{-1} \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix} \right]$ as two block matrix-vector products. Let $CA^{-1} \cdot \begin{pmatrix} \alpha_1 \\ \cdots \\ \alpha_{k/2} \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \cdot \\ \beta_{k/2} \end{pmatrix}$. Next, we compute

$$[D - CA^{-1}B]^{-1} \cdot \begin{pmatrix} \beta_1 \\ \cdots \\ \beta_{k/2} \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \cdots \\ \gamma_{k/2} \end{pmatrix}, \; (A^{-1}B) \cdot \begin{pmatrix} \gamma_1 \\ \cdots \\ \gamma_{k/2} \end{pmatrix}$$

since $[D - CA^{-1}B]^{-1}$ and $A^{-1}B$ are already calculated. Thus, computing $R^{-1} \cdot \alpha$ requires two block matrix multiplications for computing $A^{-1}B$ and $C(A^{-1}B)$, two inversions for

Table 4: Gaussian elimination (GE) vs. Block matrix inversion (BMI) method in CPU cycles.

Matrix Size	GE	BMI (Depth 1)	BMI (Depth 2)
46	94 033	72 302	—
48	101 498	75 136	72 479
50	142 243	82 768	—
56	175 195	103 357	99 445
64	225 081	87 150	70 091
68	322 315	187 947	170 788
72	355 173	208 355	190 480
96	713 462	252 538	226 627
100	923 489	453 441	391 747

A^{-1} and $[D-CA^{-1}B]^{-1}$, and four block matrix-vector products. Its complexity is reduced to $O((k/2)^3)$. Consequently, we reduce from six block matrix multiplications to two block matrix multiplications and four block matrix-vector products. □

It is known that the naive way to perform a matrix multiplication requires $O(k^3)$ operations and Strassen's algorithm for matrix multiplications requires $O(k^{log_2 7}) = O(k^{2.81})$ operations [Str69]. We do not use Strassen's algorithm for matrix multiplications to compute $R^{-1} \cdot \alpha$ since Strassen's algorithm based on 2×2 block matrices makes the implementations very slow for larger matrices due to its recursive nature, and our method requires only two block matrix multiplications.

Repeated BMI: Determine the Minimum Size of a Matrix being Inverted. We want to determine the minimum size of a matrix being inverted.

- After performing the BMI, we reduced the size of a matrix being inverted by half, while the number of inversions for the half-sized matrices increased to two. We can apply the BMI again to these two half-sized matrices which results in four inversions of $k/4 \times k/4$ matrices and extra operations.

- Like this, for $m = 2^l \cdot m'$, we can apply the BMI l times. We define the number of these iterations of the BMI as a depth. We cannot expect that l iterations will always be effective, because 2^l inversions of $k/2^l \times k/2^l$ matrices are required.

Compatibility of the BMI Method with UOV and Rainbow. Unlike defined in the Rainbow specification [DCP+20b], our BMI method computes $LS_V^{-1} \cdot \mathbf{a}$ directly without finding LS_V^{-1}. When using the BMI method for Rainbow (resp. UOV), if the block matrices of LS_V^{-1} (resp., $LS_{V,1}^{-1}$ or $LS_{V,2}^{-1}$) are not invertible then new Vinegar values need to be chosen. Thus, to apply our BMI method to the signing algorithms of UOV and Rainbow, the specification of Rainbow and UOV would need to be changed from Algorithm 2 and 5 to Algorithm 7 and 8, respectively. In Algorithm 7 and 8, $BMI(\cdot)$ takes LS_V and a vector as inputs then outputs a solution and a flag indicating LS_V is solvable similar to $Gauss(\cdot)$ in Algorithm 2. Since the loop structure for each UOV/Rainbow signature generation algorithm is changed and random number generations for Vinegar and salt are involved within the loop, resulting signatures deploying the BMI method can be different from the specification although they are valid signatures) even if the seeds for random number generators as well as documents and secret keys are identical at the initial point of the algorithms.

Algorithm 7 Rainbow Signature Generation Using the BMI Method

Require: document d, Rainbow private key ($InvS$, $\mathcal{F}$, $InvT$), length of the salt l.
Ensure: signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$ such that $\mathcal{P}(\mathbf{z}) = \mathcal{H}(\mathcal{H}(d)||r)$.

repeat
  $y_1, ..., y_{v_1} \leftarrow_R \mathbb{F}_q$
  $\hat{f}^{(v_1+1)}, ..., \hat{f^{(n)}} \leftarrow f^{(v_1+1)}(y_1, ..., y_{v_1}), ..., f^{(n)}(y_1, ..., y_{v_1})$
  $(\hat{F}, C_F) \leftarrow \texttt{Aff}^{-1}(\hat{f}^{(v+1)}, ..., \hat{f}^{(n)})$
  $r \leftarrow \{0,1\}^l$
  $\mathbf{h} \leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
  $\mathbf{x} \leftarrow InvS \cdot \mathbf{h}$
  $t_1, (y_{v_1+1}, ..., y_{v_2}) \leftarrow BMI(\hat{F}, (x_{v_1+1}, ..., x_{v_2}) - C_F)$
  $\hat{f}^{(v_2+1)}, ..., \hat{f}^{(n)} \leftarrow \hat{f}^{v_2+1}(y_{v_1+1}, ..., y_{v2}), ..., \hat{f}^{(n)}(y_{v_1+1}, ..., y_{v2})$
  $t_2, (y_{v_2+1}, ..., y_n) \leftarrow BMI((\hat{f}^{((v_2+1)}, ..., \hat{f}^{(n)}), (x_{v_2+1}, ..., x_n))$
until (t_1 == **TRUE**) & (t_2 == **TRUE**)
$\mathbf{z} = InvT \cdot \mathbf{y}$
$\sigma \leftarrow (\mathbf{z}, r)$
Return σ

Algorithm 8 UOV Signature Generation Using the BMI Method

Require: document d, UOV private key ($\mathcal{F}$, $InvT$), length of the salt l.
Ensure: signature $\sigma = (\mathbf{z}, r) \in \mathbb{F}_q^n \times \{0,1\}^l$ such that $\mathcal{P}(\mathbf{z}) = \mathcal{H}(\mathcal{H}(d)||r)$.

repeat
  $y_1, ..., y_v \leftarrow_R \mathbb{F}_q$
  $\hat{f}^{(v_1+1)}, ..., \hat{f^{(n)}} \leftarrow f^{(v+1)}(y_1, ..., y_v), ..., f^{(n)}(y, ..., y_v)$
  $(\hat{F}, C_F) \leftarrow \texttt{Aff}^{-1}(\hat{f}^{(v+1)}, ..., \hat{f}^{(n)})$
  $r \leftarrow \{0,1\}^l$
  $\mathbf{x} \leftarrow \mathcal{H}(\mathcal{H}(d)||r)$
  $t, (y_{v+1}, ..., y_n) \leftarrow BMI(\hat{F}, (x_{v+1}, ..., x_n) - C_F)$
until t == **TRUE**
$\mathbf{z} = InvT \cdot \mathbf{y}$
$\sigma \leftarrow (\mathbf{z}, r)$
Return σ

Implementation Results. We investigate improvements of UOV/Rainbow signing with the BMI method. For our implementations, we utilized Rainbow team's AVX2-specialized basic functions for GF arithmetic, basic linear algebra, and matrix operations in its source code submitted to NIST PQC Round 3. We used Rainbow team's AVX2 specialized functions for matrix operations and Gaussian elimination for the BMI implementation. The implementations of precomputation are constructed with rearranging computational operations and adjusting memory operations. We summarize their implementation results in terms of the size of matrices being inverted in Table 4 and in terms of the security categories in Table 5.

- To use the BMI with the depth 1 and the depth 2, matrix sizes should be multiples of 2 and multiples of 4, respectively. In the case of $m = 46$, we cannot use the BMI with the depth 2 since it is not a multiple of 4. We implement the BMI on the finite field $\mathbb{F}_{2^8}$. After obtaining LS_V from the Vinegar value substitution, we set $LS_V = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, if A or $[D - CA^{-1}B]$ is not invertible then we choose another Vinegar values. The probability that the matrices are invertible is 99.223%.

Table 5: UOV/Rainbow implementation results with the BMI method at three security categories in CPU cycles.

Scheme	Security Category	I	III	V
UOV	Gaussian elimination	201 834	707 959	1 486 775
	BMI (Depth 1)	176 884	563 519	1 004 704
	BMI (Depth 2)	—	535 660	981 351
Rainbow	Gaussian elimination	68 203	322 799	807 309
	BMI (Depth 1)	—	270 731	650 400
	BMI (Depth 2)	—	271 986	639 965

- As seen in Table 4 and Table 5, compared to Gaussian elimination, the larger the size, the greater the performance improvement and the higher the security category, the greater the effect.
  - By using BMI with the depth 1, we get speedups of 23.1%, 41%, 64.6%, and 50.9% at the sizes, 46, 72, 96, and 100, respectively. Especially excellent improvements of 61.3% and 64.6% in the case of 64 and 96, respectively, are due to the fact that the multiples of 32 are optimal parameters which are suitable for the AVX2 vectorization.
  - In the BMI with the depth 2, their speedups are 46.54%, 68.23%, and 57.58% at the sizes, 72, 96, and 100, respectively.
- Compared to UOV implemented with Gaussian elimination, by using the BMI with the depth 1, we obtain speedups of 12.36%, 20.41%, and 32.42% at the three security categories, respectively. For the BMI with the depth 2, the corresponding speedups are 24.3% and 34% at the security categories III and V, respectively. Compared to the BMI with the depth 1, the effect of the BMI with the depth 2 is insignificant.
- Compared to Rainbow implemented with Gaussian elimination, by using BMI with the depth 1, we obtain speedups 16.13% and 19.44% at the security categories III and V, respectively. For the BMI with depth 2, the corresponding speedup is 15.74% and 20.72 % at the security categories III and V, respectively. Our results show that the BMI with larger depths are not always more efficient. Our BMI method in Rainbow implementation at the security category I is not effective. Since its parameter $(\mathbb{F}_{2^4}, 36, 32, 32)$ is very optimal, the implementation with Gaussian elimination is faster than that with the BMI method.

4.2 Splitting Signing into Offline and Online Phase

To speed up the signing process even more, we can choose to split the signature generation in an offline and online phase, where the offline phase can already be performed before the message to be signed is known. In 1989, Even, Goldreich, and Micali [EGM90] introduced the general idea of using an offline/online phase. Most costly message independent computations are completed in the offline phase and the device is idle. Such precomputation enables the online phase to quickly sign the messages with only light computation. Now, we describe the precomputation parts of UOV/Rainbow signing and evaluate the performance of offline/online computations using the AVX2 instruction set.

4.2.1 UOV with Precomputation

Unlike Rainbow, UOV has significantly large message independent operations in signing. This offline precomputation can dramatically improve signing in UOV.

Signing. Signing can be divided into two parts: one is independent of messages being signed, the other depends on the messages.

- After substituting $s_V = (s_1, \cdots, s_v)$ into o equations $\mathcal{F}^{(k)}$ $(1 \leq k \leq o)$, one gets a linear system of o equations and o variables $x_{v+1}, \cdots, x_{v+o}$ and computes LS_V^{-1}. Thus, $< s_V = (s_1, \cdots, s_V),\ c_V = (c_1, \cdots, c_m),\ LS_V^{-1} >$ can be precomputed, where c_V is a vector of constant terms of $(\mathcal{F}^{(1)}(s_V), \cdots, \mathcal{F}^{(m)}(s_V))$.
- This precomputation speedups the signing performance. However, it requires an additional computational overhead and one has to decide when to calculate and where to keep precomputed values. We will discuss how to use the precomputed values securely in the next subsection.

Offline Signing of UOV.

- After choosing random Vinegar values $s_V = (s_1, \cdots, s_v) \in \mathbb{F}_q^v$, substitute s_V into o equations $\mathcal{F}^{(k)}$ $(1 \leq k \leq o)$ to get the linear system LS_V of o equations and o unknowns and a constant vector $c_V = (c_1, \cdots, c_m)$.
- Compute LS_V^{-1}. If LS_V is not invertible then go back to the first step.
- Store $< s_V, c_V, LS_V^{-1} >$ as the precomputed values.

Online Signing of UOV.

- Choose a random salt r and compute $h = \mathcal{H}(\mathcal{H}(\mathtt{m})||r)$ for a message $\mathtt{m}$.
- From $< s_V = (s_1, \cdots, s_V),\ c_V = (c_1, \cdots, c_m),\ LS_V^{-1} >$, compute $LS_V^{-1} \cdot h_V^{\mathtt{T}} = \alpha$, where $h_V = (h_1 - c_1, \cdots, h_m - c_m)$ and $h = (h_1, \cdots, h_m)$.
- Compute $T^{-1} \cdot (S_V, \alpha)^{\mathtt{T}} = \sigma$ and output $\tau = (\sigma, r)$ as a signature on $\mathtt{m}$.

4.2.2 Rainbow with Precomputation

In the first layer of Rainbow, the same precomputation is possible, but in the second layer, precomputation is limited since the vector consisting of the random Vinegar values and the solution of the linear system related to the message is supposed to be substituted into the central polynomials of the second layer. Hence, with the precomputed values for the first layer and some possible part of the second layer, computing a solution of the linear system from the first layer and then substituting the solution to the central polynomials and solving the linear system in the second layer are required for the online phase.

Offline Signing of Rainbow.

- After choosing random Vinegar values $s_V = (s_1, \cdots, s_v) \in \mathbb{F}_q^v$, substitute s_V into o_1 equations $\mathcal{F}^{(k)}$ $(1 \leq k \leq o_1)$ to get the linear system $LS_{V,1}$ of o_1 equations and o_1 unknowns and a vector $c_{V,1} = (c_1, \cdots, c_{o_1})$ of constant terms $(\mathcal{F}^{(1)}(s_V), \cdots, \mathcal{F}^{(o_1)}(s_V))$.
- Compute $LS_{V,1}^{-1}$. If $LS_{V,1}$ is not invertible then go back to the first step.
- Substitute s_V into the Vinegar variables in o_2 equations $\mathcal{F}^{(k)}$ $(o_1 + 1 \leq k \leq o_1 + o_2)$ by getting a linear system of o_2 equations and unknowns, $\{\mathcal{F}^{(o_1+i)}(s_V)\}_{i=1}^{o_2}$.

- Store $< s_{V,1},\ c_{V,1},\ LS_{V,1}^{-1},\ \{\mathcal{F}^{(o_1+i)}(s_V)\}_{i=1}^{o_2} >$.

Online Signing of Rainbow.

- Compute $S^{-1} \cdot h^{\mathtt{T}} = \alpha$ for $h = \mathcal{H}(\mathcal{H}(m)||r)$ with a message $\mathtt{m}$ and a random salt r.
- From $< s_{V,1}, c_{V,1}, LS_{V,1}^{-1}, \{\mathcal{F}^{(o_1+i)}(s_V)\}_{i=1}^{o_2} >$, compute $LS_{V,1}^{-1} \cdot \alpha^{\mathtt{T}} = \delta$, where $\alpha = (\alpha_1 - c_1, \cdots, \alpha_{o_1} - c_{o_1})$ and $\alpha = (\alpha_1, \cdots, \alpha_{o_1+o_2})$.
- Compute $LS_{V,2}^{-1}$, where $LS_{V,2}$ is the coefficient matrix of the linear system obtained by substituting α into o_1-oil variables in $\{\mathcal{F}^{(o_1+i)}(s_V, \delta)\}_{i=1}^{o_2}$. If $LS_{V,2}$ is not invertible then select another salt and go back to the first step.
- Compute $LS_{V,2}^{-1} \cdot (\alpha_{o_1+1}, \cdots, \alpha_{o_1+o_2})^{\mathtt{T}} = \zeta$ and $T^{-1} \cdot (S_V, \delta, \zeta)^{\mathtt{T}} = \sigma$, and output $\tau = (\sigma, r)$ as a signature on $\mathtt{m}$.

Implementation Results. Table 6 lists the benchmarking results of our UOV/Rainbow implementations with and without precomputation in cycle counts and required additional memory cost per signature.

- In Table 6, Sign w/ Precomp. and Sign w/o Precomp. represent signing with precomputation and signing without precomputation, respectively. Precomp. Memory Cost per Sig. represents the memory cost required for precomputation per signature.
- Compared to UOV without precomputation, UOV with precomputation obtains speedups of 94%, 97%, and 98.43% at the three security categories, respectively.
- In Rainbow, the large portion of the second layer operations have to be done in the online phase. Compared to Rainbow without precomputation, Rainbow with precomputation obtains speedups of 52.89%, 55.16%, and 64.32% at the three security categories, respectively.
- UOV with precomputation is about 2.7 times, 7.3 times and 12.2 times faster than Rainbow with precomputation at the three security categories, respectively.
- The results of Table 6 are without using the BMI method. The BMI method and precomputation can not be used together. The goal of the BMI method is to compute $LS_V^{-1} \cdot \alpha$, not LS_V^{-1}. A message is needed to compute $LS_V^{-1} \cdot \alpha$, where LS_V is determined by the Vinegar values and a vector α is determined by a message being signed. Thus, to use the BMI method, the message is required, in that case, precomputation for message-independent operations is not possible.

Signing of Rainbow is very fast, but the higher the security level, the slower the performance. Thus, precomputation is suitable for all applications where maintaining fast performance is required even at the high security levels. Also, precomputation-based schemes fit into applications deploying the nodes with cost-effective energy harvesting technologies. The cost-effective energy harvesting technologies in WSNs are devised to supplement the battery power with energy gathered from the environment (e.g. solar, wind) and energy peaks are occasionally available. In some cases, the available energy can be even excessive, i.e. greater than the amount that can fit into the sensor node's supercapacitor, and thus it would be wasted unless used immediately [WWCJ10]. In this case, precomputation can be done for efficient use of energy.

Compared to recent precomputation results of Dilithium (lattice-based post-quantum signature scheme) in [RGCB19], Dilithium 3 with precomputation speedups upto 35% and requires an additional 260 KiB of space for the precomputed values per signature. Furthermore, in Dilithium's precomputation, there was a 95% probability that at least one of the y values results in a good signature.

Table 6: UOV/Rainbow implementations with and without precomputation at three security categories, in CPU cycles and bytes.

Scheme	Security Category	Unit	I	III	V
UOV	Sign w/o Precomp.	cycle	201 834	707 959	1 486 775
	Precomp. (offline)		189 224	690 586	1 460 168
	Sign w/ Precomp. (online)	cycle	11 788	19 439	23 133
	Total (offline + online)		201 012	710 025	1 483 301
	Precomp. Memory Cost per Sig.	byte	2 256	5 402	9 504
Rainbow	Sign w/o Precomp.	cycle	68 203	322 799	807 309
	Precomp. (offline)		37 212	173 204	508 890
	Sign w/ Precomp. (online)	cycle	31 973	142 179	278 511
	Total (offline + online)		69 185	315 383	787 401
	Precomp. Memory Cost per Sig.	byte	2 152	4 792	5 648

4.3 Resilience against Leakage or Reuse of Precomputed Values

We have shown that precomputation dramatically improved UOV/Rainbow signing with the additional memory cost. To use the precomputed values securely, we investigate the resilience against leakage or reuse of the precomputed values in UOV and Rainbow. We show that a sufficient number of signatures generated by the precomputed values lead the full secret key recoveries of UOV and Rainbow in polynomial-time.

4.3.1 Leakage of Precomputed Values

Store $< s_V, c_V, LS_V^{-1} >$ Securely. The precomputed values $< s_V, c_V, LS_V^{-1} >$ should be stored securely. If some precomputed values together with signatures generated by them are exposed then the secret key of UOV is completely recovered.

Wolf and Preneel [WP05] introduced the notion of equivalent keys as a fundamental tool to analyze the security of the MQ-schemes. They showed that there exist a large number of different secret keys (called equivalent keys) for a given public key in UOV and Rainbow [WP05, Tho13]. If an attacker finds any equivalent keys then it can forge any signatures on any messages without acquiring the original secret key. Key recovery attacks (KRAs) exploit the special structure of the central map, i.e. quadratic terms with zero coefficients at certain known places. Goal of the KRAs is to find an equivalent key associated to the public key. In the following theorems, we show that the leakage of precomputed values and $(n+1)$ signatures generated by the precomputed values leads to the recovery of the equivalent keys of UOV and Rainbow in polynomial-time. They can be applied to both cases of random affine secret keys and equivalent keys.

Theorem 3. If $(n+1)$ tuples $< \mathtt{m}^{(i)}, \tau^{(i)}, s_V^{(i)}, \ c_V^{(i)}, LS_V^{(i)-1} >$ are given such that the $n \times n$ matrix $(\sigma^{(1)\mathtt{T}} \ \sigma^{(2)\mathtt{T}} \ \cdots \ \sigma^{(n)\mathtt{T}})$ is invertible then the secret key of UOV is completely recovered in polynomial-time.

Proof. Let T be the linear part of $\mathcal{T}$ and $\mathcal{T}_V = \pi_v \circ \mathcal{T} : \mathbb{F}_q^n \mapsto \mathbb{F}_q^v$ and $T_v = \pi_v \circ T : \mathbb{F}_q^n \mapsto \mathbb{F}_q^v$, where $\pi_v(x_1, \cdots, x_v, \cdots, x_n) = (x_1, \cdots, x_v)$. By assumption, we get $T_V(\sigma^{(i)} - \sigma^{(j)}) = \mathcal{T}_V(\sigma^{(i)}) - \mathcal{T}_V(\sigma^{(j)}) = s_V^{(i)} - s_V^{(j)}$. For each $1 \le i \le n$, let $\sigma_i = \sigma^{(i)} - \sigma^{(n+1)}$ and

$\hat{s_V}^{(i)} = s_V^{(i)} - s_V^{(n+1)}$. Then we get $T_V(\sigma_i) = \hat{s_V}^{(i)}$ and

$$T_V \cdot \left(\sigma_1^{\mathsf{T}} \;\; \sigma_2^{\mathsf{T}} \;\; \cdots \;\; \sigma_n^{\mathsf{T}}\right) = \left(\hat{s_V}^{(1)\mathsf{T}} \;\; \hat{s_V}^{(2)\mathsf{T}} \;\; \cdots \;\; \hat{s_V}^{(n)\mathsf{T}}\right)$$

Since an n-dimensional diagonal matrix $(\sigma_1^{\mathsf{T}} \;\; \sigma_2^{\mathsf{T}} \;\; \cdots \;\; \sigma_n^{\mathsf{T}})$ is invertible with high probability $\left(1 - \frac{1}{q}\right)^n$, we get

$$T_V = \left(\hat{s_V}^{(1)\mathsf{T}} \;\; \hat{s_V}^{(2)\mathsf{T}} \;\; \cdots \;\; \hat{s_V}^{(n)\mathsf{T}}\right) \cdot \left(\sigma_1^{\mathsf{T}} \;\; \sigma_2^{\mathsf{T}} \;\; \cdots \;\; \sigma_n^{\mathsf{T}}\right)^{-1}.$$

It is known that there is an equivalent key $(\mathcal{F}', \mathcal{T}')$ with $\mathcal{P} = \mathcal{F}' \circ \mathcal{T}'$ where $\mathcal{F}'$ has the same structure with $\mathcal{F}$ and linear coefficient matrix T' of $\mathcal{T}'$ has the following simple form

$$T' = \begin{pmatrix} I^{v\times v} & T_0'^{v\times o} \\ 0^{o\times v} & I^{o\times o} \end{pmatrix}.$$

Note that the first v rows of T' can be obtained by a reduced row echelon form T_V. Therefore, an equivalent key of UOV is recovered in polynomial time. □

For Rainbow, we consider its equivalent keys. Let

$$S^{-1} = \begin{pmatrix} \widetilde{S_1}^{o_1\times o_1} & \widetilde{S_2}^{o_1\times o_2} \\ \widetilde{S_3}^{o_2\times o_1} & \widetilde{S_4}^{o_2\times o_2} \end{pmatrix}, \quad T^{-1} = \begin{pmatrix} \widetilde{T_1}^{v\times v} & \widetilde{T_2}^{v\times o_1} & \widetilde{T_3}^{v\times o_2} \\ \widetilde{T_4}^{o_1\times v} & \widetilde{T_5}^{o_1\times o_1} & \widetilde{T_6}^{o_1\times o_2} \\ \widetilde{T_7}^{o_2\times v} & \widetilde{T_8}^{o_2\times o_1} & \widetilde{T_9}^{o_2\times o_2} \end{pmatrix},$$

S and T are linear part of $\mathcal{S}$ and $\mathcal{T}$, respectively. For a secret key $(\mathcal{F}, \mathcal{S}, \mathcal{T})$ of Rainbow, if there exists

$$(\mathcal{F}', \mathcal{S}', \mathcal{T}') = (\Sigma \circ \mathcal{F} \circ \Omega,\ \mathcal{S} \circ \Sigma^{-1},\ \Omega^{-1} \circ \mathcal{T}),$$

and $\Sigma \circ \mathcal{F} \circ \Omega$ and $\mathcal{F} \circ \Omega$ have the same structure for some linear maps $\Sigma \in \mathbb{GL}_m(\mathbb{F}_q)$ and $\Omega \in \mathbb{GL}_n(\mathbb{F}_q)$, then $(\mathcal{F}', \mathcal{S}', \mathcal{T}')$ is an equivalent key of $(\mathcal{F}, \mathcal{S}, \mathcal{T})$, where Σ and Ω are of the form

$$\Sigma = \begin{pmatrix} \Sigma_1^{o_1\times o_1} & 0^{o_1\times o_2} \\ \Sigma_2^{o_2\times o_1} & \Sigma_3^{o_2\times o_2} \end{pmatrix}, \quad \Omega = \begin{pmatrix} \Omega_1^{v\times v} & 0^{v\times o_1} & 0^{v\times o_2} \\ \Omega_2^{o_1\times v} & \Omega_3^{o_1\times o_1} & 0^{o_1\times o_2} \\ \Omega_4^{o_2\times v} & \Omega_5^{o_2\times o_1} & \Omega_6^{o_2\times o_2} \end{pmatrix}. \tag{1}$$

If we set submatrices of Σ and Ω in (1) as

$$\Sigma_1 = \left(\widetilde{S_1}\right)^{-1}, \quad \begin{pmatrix} \Sigma_2 & \Sigma_3 \end{pmatrix} = \begin{pmatrix} 0^{o_2\times o_1} & I^{o_2\times o_2} \end{pmatrix} \cdot S,$$
$$\begin{pmatrix} \Omega_1 \\ \Omega_2 \\ \Omega_4 \end{pmatrix} = T \cdot \begin{pmatrix} I^{v\times v} \\ 0^{m\times v} \end{pmatrix}, \quad \begin{pmatrix} \Omega_3 \\ \Omega_5 \end{pmatrix} = \begin{pmatrix} \widetilde{T_5} & \widetilde{T_6} \\ \widetilde{T_8} & \widetilde{T_9} \end{pmatrix}^{-1} \begin{pmatrix} I^{o_1\times o_1} \\ 0^{o_2\times o_1} \end{pmatrix}, \quad \Omega_6 = \left(\widetilde{T_9}\right)^{-1},$$

one gets

$$\begin{aligned} S'^{-1} &= \Sigma \cdot S^{-1} = \begin{pmatrix} I^{o_1\times o_1} & \widetilde{S_1'}^{o_1\times o_2} \\ 0^{o_2\times o_1} & I^{o_2\times o_2} \end{pmatrix}, \\ T'^{-1} &= T^{-1} \cdot \Omega = \begin{pmatrix} I^{v\times v} & \widetilde{T_1'}^{v\times o_1} & \widetilde{T_2'}^{v\times o_2} \\ 0^{o_1\times v} & I^{o_1\times o_1} & \widetilde{T_3'}^{o_1\times o_2} \\ 0^{o_2\times v} & 0^{o_2\times o_1} & I^{o_2\times o_2} \end{pmatrix}, \end{aligned} \tag{2}$$

where S' and T' are coefficient matrices of linear part of $\mathcal{S}'$ and $\mathcal{T}'$, respectively [Tho13].

Theorem 4. If $(n+1)$ tuples $< \mathtt{m}^{(i)}, \tau^{(i)}, s_V^{(i)},\ c_V^{(i)}, (LS_V^{(i)})^{-1} >$ are given such that the $n \times n$ matrix $(\sigma^{(1)\mathtt{T}}\ \sigma^{(2)\mathtt{T}}\ \cdots\ \sigma^{(n)\mathtt{T}})$ is invertible then an equivalent key of Rainbow is completely recovered in polynomial-time.

Proof. Let $P = \left(P^{(1)}, P^{(2)}, \cdots, P^{(m)}\right)$ and $F = \left(F^{(1)}, F^{(2)}, \cdots, F^{(m)}\right)$ be the quadratic part of $\mathcal{P}$ and $\mathcal{F}$, respectively, and S and T are the linear part of $\mathcal{S}$ and $\mathcal{T}$, respectively. Then we have $P = S \circ F \circ T$. We denote

$$O_0 = \{(x_1, \cdots, x_n) \in \mathbb{F}_q^n : x_1 = \cdots = x_v = 0\},$$
$$O_1 = \{(x_1, \cdots, x_m) \in \mathbb{F}_q^n : x_1 = \cdots = x_{o_1} = 0\}.$$

We can see that if $x \in O_0$ then $F^{(k)}(x) = 0$ for $1 \le k \le o_1$ and hence $F(x) \in O_1$.
Let $T_v = \pi_v \circ T : \mathbb{F}_q^n \mapsto \mathbb{F}_q^v$, where $\pi_v(x_1, \cdots, x_v, \cdots, x_n) = (x_1, \cdots, x_v)$. Similarly with the proof of Theorem 1, we can recover T_V if we have $n+1$ tuples in assumption. Let Λ be the orthogonal space of the space generated by row vectors of T_V. Then if $\mathbf{b} \in \Lambda$ we get $T_V(\mathbf{b}) = 0$ and hence we get $T(\mathbf{b}) \in O_0$. So, we get $S^{-1} \circ P(\mathbf{b}) = (F \circ T)(\mathbf{b}) \in O_1$. We choose o_2 vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{o_2} \in \Lambda$ and denote $\mathbf{a}_i = P(\mathbf{b}_i) \in S(O_1)$ for each $1 \le i \le o_2$. Then $S^{-1}(\mathbf{a}_i) \in O_1$. If $\mathbf{s}_j$ be j-th row vector of S^{-1} for $1 \le j \le m$ then we can see that $\mathbf{a}_i \cdot \mathbf{s}_j = 0$ for every $1 \le i \le o_2$ and $1 \le j \le o_1$. Let $\mathtt{S}$ be an o_1-dimensional space generated by $\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{o_1}$. For each $\mathtt{s} = (\mathtt{s}_1, \cdots, \mathtt{s}_m) \in \mathtt{S}$, we get o_2 linear equations of the form

$$\mathbf{a}_i \cdot (\mathtt{s}_1, \cdots, \mathtt{s}_m) = 0$$

about m variables $\mathtt{s}_1, \cdots, \mathtt{s}_m$. We solve these linear equations, and express $\mathtt{s}_{o_1+1}, \cdots, \mathtt{s}_m$ by linear combinations of $\mathtt{s}_1, \cdots, \mathtt{s}_{o_1}$ such as

$$\begin{pmatrix} \mathtt{s}_1 \\ \mathtt{s}_2 \\ \vdots \\ \mathtt{s}_{o_1} \end{pmatrix} = \begin{pmatrix} \widetilde{s}_{1,o_1+1} \\ \widetilde{s}_{2,o_1+1} \\ \vdots \\ \widetilde{s}_{o_1,o_1+1} \end{pmatrix} \mathtt{s}_{o_1+1} + \begin{pmatrix} \widetilde{s}_{1,o_1+2} \\ \widetilde{s}_{2,o_1+2} \\ \vdots \\ \widetilde{s}_{o_1,o_1+2} \end{pmatrix} \mathtt{s}_{o_1+2} + \cdots + \begin{pmatrix} \widetilde{s}_{1,m} \\ \widetilde{s}_{2,m} \\ \vdots \\ \widetilde{s}_{o_1,m} \end{pmatrix} \mathtt{s}_m$$
$$= \begin{pmatrix} \widetilde{s}_{1,o_1+1} & \widetilde{s}_{1,o_1+2} & \cdots & \widetilde{s}_{1,m} \\ \widetilde{s}_{2,o_1+1} & \widetilde{s}_{2,o_1+2} & \cdots & \widetilde{s}_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{s}_{o_1,o_1+1} & \widetilde{s}_{o_1,o_1+2} & \cdots & \widetilde{s}_{o_1,m} \end{pmatrix} \begin{pmatrix} \mathtt{s}_{o_1+1} \\ \mathtt{s}_{o_1+2} \\ \vdots \\ \mathtt{s}_m \end{pmatrix} = \widetilde{S} \begin{pmatrix} \mathtt{s}_{o_1+1} \\ \mathtt{s}_{o_1+2} \\ \vdots \\ \mathtt{s}_m \end{pmatrix}.$$

Thus, $\mathtt{S}$ is generated by row vectors of

$$\left(I^{o_1 \times o_1} \quad \widetilde{S}\right).$$

Since the reduced row echelon form is unique, we can see that

$$\begin{pmatrix} I^{o_1 \times o_1} & \widetilde{S} \\ 0^{o_2 \times o_1} & I^{o_2 \times o_2} \end{pmatrix} = S'^{-1},$$

where S'^{-1} is in (2). Hence, we can find S'^{-1} in (2) in polynomial-time. The remaining part of the attack is to recover T'^{-1} in (2). We show that we can recover T'^{-1} by solving several linear systems. For the last o_2 columns of T'^{-1}, we can apply good key finding attack in [Tho13]. More precisely, one can find a good key $(\mathcal{F}' \circ \Omega', \mathcal{S}', \mathcal{T}'')$ corresponding to $(\mathcal{F}', \mathcal{S}', \mathcal{T}')$, where T'', the coefficient matrix of $\mathcal{T}''$, satisfies

$$T''^{-1} = T'^{-1} \cdot \Omega' = \begin{pmatrix} I^{v \times v} & 0^{v \times o_1} & 0^{v \times (o_2-1)} & \widetilde{T_2''}^{v \times 1} \\ 0^{o_1 \times v} & I^{o_1 \times o_1} & 0^{o_1 \times (o_2-1)} & \widetilde{T_3''}^{o_1 \times 1} \\ 0^{(o_2-1) \times v} & 0^{(o_2-1) \times o_1} & I^{(o_2-1) \times (o_2-1)} & 0^{(o_2-1) \times 1} \\ 0^{1 \times v} & 0^{1 \times o_1} & 0^{1 \times (o_2-1)} & 1 \end{pmatrix},$$

if we set

$$\Omega' = \begin{pmatrix} T'_{n,n-1} & 0^{(n-1)\times 1} \\ & 1 \end{pmatrix},$$

where $T'_{n,n-1}$ is the $n \times (n-1)$ submatrix of T' removing the last column from T'. Note that $\begin{pmatrix} \widetilde{T''_2} \\ \widetilde{T''_3} \end{pmatrix}$ is identical to the $(v+o_1) \times 1$ submatrix of $\begin{pmatrix} \widetilde{T'_2} \\ \widetilde{T'_3} \end{pmatrix}$ consisting of the last column. Then we get a multivariate system to find this equivalent key that consists of the following form of equations

$$F''^{(k)}_{i,j} = \sum_{a=1}^{m}\sum_{b=1}^{n}\sum_{c=b}^{n} P^{(a)}_{b,c}\widetilde{s''_{k,a}}\widetilde{t''_{b,i}}\widetilde{t''_{c,j}},$$

where $F''^{(k)} = \left(F''^{(k)}_{i,j}\right)_{i,j}$ is the symmetric coefficient matrix of quadratic part of $\mathcal{F}''^{(k)}$ for $1 \le k \le m$ and $S''^{-1} = \left(\widetilde{s''_{i,j}}\right)_{i,j}, T''^{-1} = \left(\widetilde{t''_{i,j}}\right)_{i,j}$. We get $F''^{(k)}_{i,n} = 0$ for $1 \le i \le n-1$ and $1 \le k \le o_1$. More precisely, we get $o_1(n-1)$ equations of the following form

$$F''^{(k)}_{i,n} = \sum_{a=o_1+1}^{m}\sum_{b=1}^{v+o_1} P^{(a)}_{b,n}\widetilde{s'_{k,a}}\widetilde{t'_{b,n}} + \sum_{a=o_1+1}^{m} P^{(a)}_{n,n}\widetilde{s'_{k,a}} + \sum_{b=1}^{v+o_1} P^{(k)}_{b,n}\widetilde{t'_{b,n}} + P^{(k)}_{n,n}.$$

Since we have already recovered all entries of S'^{-1}, all obtained equations are linear about $\widetilde{t'_{b,n}}$. So, we have a linear system of $o_1(n-1)$ equations with $v+o_1$ variables which can be solvable in polynomial time. Now, we need to recover $\widetilde{T'_1}$ in (2). We consider equations of the form

$$F'^{(k)}_{i,j} = \sum_{a=1}^{m}\sum_{b=1}^{n}\sum_{c=b}^{n} P^{(a)}_{b,c}\widetilde{s'_{k,a}}\widetilde{t'_{b,i}}\widetilde{t'_{c,j}}$$

which can be obtained from $F = S'^{-1} \circ P \circ T'^{-1}$, where $v+o_1+1 \le j \le n$ and $1 \le k \le o_1$. Since we already recover all entries of S'^{-1} and $\widetilde{t'_{c,j}}$, we get $o_1 o_2$ linear equations with v variables. For suggested parameters, we get enough equations to recover these variables. Therefore, we can totally recover an equivalent key by solving several linear systems. □

4.3.2 Reuse of Precomputed Values

Do not Reuse $< s_V, c_V, LS_V^{-1} >$**.** The precomputed value $< s_V, c_V, LS_V^{-1} >$ should not be reused in signing.

Recently, Shim and Koo [SK20] showed that the equivalent key of UOV is completely recovered in polynomial-time from $(m+1)$ signatures generated by the reused Vinegar values. However, they did not show a polynomial-time recovery of Rainbow's equivalent key. Their results are given the following Theorem 5.

Theorem 5. **[SK20]** If $(m+1)$ signatures generated by the reused Vinegar values are given then

- the equivalent key of UOV is completely recovered in polynomial time,
- the complexity of the KRAs using good keys on Rainbow is determined by solving a multivariate system of m quadratic equations with o_1 variables.

Now, we provide a more improved analysis: (o_2+1) signatures generated by the fixed Vinegar values lead to the full secret key recovery of Rainbow in polynomial-time.

Theorem 6. If $(o_2 + 1)$ signatures generated by reusing the precomputed values then an equivalent key of Rainbow is recovered in polynomial-time with high probability.

Proof. As in Theorem 2, we define $P = \left(P^{(1)}, P^{(2)}, \cdots, P^{(m)}\right)$, $F = \left(F^{(1)}, F^{(2)}, \cdots, F^{(m)}\right)$, S, T, O_0, and O_1. We can see that if $x \in O_0$ then $F^{(k)}(x) = 0$ for $1 \leq k \leq o_1$ and thus $F(x) \in O_1$. Suppose that $(o_2 + 1)$ signatures $(\sigma^{(1)}, \cdots, \sigma^{(o_2+1)})$ are generated by the precomputed values. The probability of getting $(o_2 + 1)$ signatures generated by the fixed Vinegar vector s_V is about

$$\left(1 - \frac{1}{q}\right)^{o_2}.$$

Define $T_v = \pi_v \circ T : \mathbb{F}_q^n \mapsto \mathbb{F}_q^v$, where $\pi_v(x_1, \cdots, x_v, \cdots, x_n) = (x_1, \cdots, x_v)$. Then, we get $T_v(\sigma^{(i)}) = T_v(\sigma^{(j)})$ for $1 \leq i < j \leq o_2 + 1$ with high probability. Since T_v is a linear transformation, we get

$$T_v(\sigma^{(i)} - \sigma^{(j)}) = T_v(\sigma^{(i)}) - T_v(\sigma^{(j)}) = 0,$$

for $1 \leq i < j \leq o_2 + 1$. Define $\mathbf{b}_i = \sigma^{(1)} - \sigma^{(i+1)}$ for $1 \leq i \leq o_2$. Then $T_v(\mathbf{b}_i) = 0$ for $1 \leq i \leq m$, and $T(\mathbf{b}_i) \in O_0$. Thus, $(F \circ T)(\mathbf{b}_i) \in O_1$ and $S^{-1} \circ P(\mathbf{b}_i) \in O_1$ for all $1 \leq i \leq o_2$. Let $\mathbf{a}_i = P(\mathbf{b}_i) \in S(O_1)$ for each $1 \leq i \leq o_2$, then $S^{-1}(\mathbf{a}_i) \in O_1$. If $\mathbf{s}_j$ be the j-th row vector of S^{-1} for $1 \leq j \leq m$ then we can see that $\mathbf{a}_i \cdot \mathbf{s}_j = 0$ for every $1 \leq i \leq o_2$ and $1 \leq j \leq o_1$. Let $\mathtt{S}$ be an o_1-dimensional space generated by $\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_{o_1}$. For each $\mathtt{s} = (\mathtt{s}_1, \cdots, \mathtt{s}_m) \in \mathtt{S}$, we get o_2 linear equations of the form

$$\mathbf{a}_i \cdot (\mathtt{s}_1, \cdots, \mathtt{s}_m) = 0$$

about m variables $\mathtt{s}_1, \cdots, \mathtt{s}_m$. The rest of this proof is the same as that of Theorem 2. Therefore, an equivalent key of Rainbow is completely recovered in polynomial-time. □

Note that the results presented in the above theorems are determined by leakage or reused Vinegar values together with signatures generated by the Vinegar values. Other values in the precomputed values except the Vinegar values do not affect the attacks in the theorems.

Zambonin *et al.*'s Variants of Rainbow. At Africacrypt 2019, Zambonin *et al.* [ZBC19] proposed three variants of Rainbow to shorten the secret key size by using fixed Vinegar values and central polynomials substituted by the fixed Vinegar values in key generations. In other words, the secret key includes the fixed Vinegar values and all the signatures are generated by the same Vinegar values in their scheme. Theorem 6 shows that their three variants are entirely broken by the key recovery attacks using good keys.

Applicability of Our Optimization to Cyclic/Compressed Versions of UOV and Rainbow. The signing processes of Cyclic versions of UOV and Rainbow are identical to those of the original versions. The signing processes of their Compressed versions are identical to those of the original versions except that the Compressed versions require additional secret key recoveries before generating a signature. Thus, our BMI method and precomputation can be applied to signing in Cyclic/Compressed versions of UOV and Rainbow.

Comparison. We summarize the overall overview of UOV/Rainbow including our optimizations in Table 7. In the Table, Comp., Precomp., Mem., M-V prod., subst., Inv., and PV are abbreviations of computation precomputation, memory, matrix-vector product, substitution, inversion, and precomputed values, respectively. Online Comp. and Mem. Cost represents the computation required to generate a signature with the precomputed values in online phase and the memory cost required for precomputation per signature, respectively.

Table 7: Comparison of UOV/Rainbow with our optimizations.

Scheme			UOV	Rainbow
Structure			ASA, Single Layer	ASA, Two Layers
Public Key			$\mathcal{P} = \mathcal{F} \circ \mathcal{T}$	$\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$
Hard Problems			MQ, EIP	MQ, EIP, MinRank
Invert			Oil-Vinegar Method	Oil-Vinegar Method
Solving Linear Systems	Sign (GE)	I	201 834 cycles	68 203 cycles
		III	707 959 cycles	322 799 cycles
		V	1 486 775 cycles	807 309 cycles
	Sign (BMI)	I	176 884 cycles	—
		III	563 519 cycles	270 731 cycles
		V	981 351 cycles	650 400 cycles
Precomp.	PV		(s_V, c_V, LS_V^{-1})	$(s_{V,1}, c_{V,1}, LS_{V,1}^{-1}, \{\mathcal{F}^{(o_1+i)}(s_V)\}_{i=1}^{o_2})$
	Online Comp.		Two M-V prod.	Subst. of o_1-values, One Inv., Three M-V prod.
	Mem. Cost		$2v + m^2$ bytes	$2v + o_1^2 + (o_2+1)o_2$ bytes
	Sign w/ Precomp.	I	11 968 cycles	32 129 cycles
		III	19 968 cycles	144 735 cycles
		V	23 667 cycles	288 008 cycles
Resistant	PV Leakage		Insecure $(n+1)$	Insecure $(n+1)$
	PV Reuse		Insecure $(m+1)$	Insecure (o_2+1)

5 Conclusion

We present efficient implementations of UOV and Rainbow by using new optimizations for signing to accelerate solving linear systems and the Vinegar value substitution. In our optimizations, the larger the size of a matrix being inverted, the greater the performance improvement and the higher the security level, the greater the effect of the optimizations. Precomputations of UOV and Rainbow improve their signing performance. The secure use of the precomputation methods requires careful implementations that do not reuse the Vinegar values and do not expose the precomputed values. The potential risks can be prevented by deleting the Vinegar values and the precomputed values immediately after using them.

References

[ABC+20] Martin R. Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Kenneth G. Paterson, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wen Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[BBB+20] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An algebraic attack on rank metric code-based cryptosystems. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 64–93. Springer, Heidelberg, May 2020.

[BBC+20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. 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.

[BERW08] Andrey Bogdanov, Thomas Eisenbarth, Andy Rupp, and Christopher Wolf. Time-area optimized public-key engines: Cryptosystems as replacement for elliptic curves? In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES 2008*, volume 5154 of *LNCS*, pages 45–61. Springer, Heidelberg, August 2008.

[Beu20] Ward Beullens. Improved cryptanalysis of UOV and rainbow. Cryptology ePrint Archive, Report 2020/1343, 2020. https://eprint.iacr.org/2020/1343.

[CCC+09] Anna Inn-Tung Chen, Ming-Shing Chen, Tien-Ren Chen, Chen-Mou Cheng, Jintai Ding, Eric Li-Hsiang Kuo, Frost Yu-Shuang Lee, and Bo-Yin Yang. SSE implementation of multivariate PKCs on modern x86 CPUs. In Christophe Clavier and Kris Gaj, editors, *CHES 2009*, volume 5747 of *LNCS*, pages 33–48. Springer, Heidelberg, September 2009.

[CDH+20] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRU. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[CHT12] Peter Czypek, Stefan Heyse, and Enrico Thomae. Efficient implementations of MQPKS on constrained devices. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 374–389. Springer, Heidelberg, September 2012.

[DCP+19] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[DCP+20a] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Modified parameters of Rainbow in response to a refined analysis of the Rainbow band separation attack by the NIST team and the recent new MinRank attacks. Technical report, Rainbow Team, 2020. available at https://troll.iis.sinica.edu.tw/by-publ/recent/rainbow-pars.pdf.

[DCP+20b] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[DCP+20c] Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, Bo-Yin Yang, Matthias Kannwischer, and Jacques Patarin. Response to recent paper by Ward Beullens. Technical report, Rainbow Team, 2020. available at https://troll.iis.sinica.edu.tw/by-publ/recent/response-ward.pdf.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 164–175. Springer, Heidelberg, June 2005.

[EGM90] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital schemes. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 263–275. Springer, Heidelberg, August 1990.

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

[LDK+20] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[MI88] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynominal-tuples for efficient signature-verification and message-encryption. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 419–453. Springer, Heidelberg, May 1988.

[ND+77] Ben Noble, James W Daniel, et al. *Applied linear algebra*, volume 3. Prentice-Hall Englewood Cliffs, NJ, 1977.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 33–48. Springer, Heidelberg, May 1996.

[PCG01] Jacques Patarin, Nicolas Courtois, and Louis Goubin. QUARTZ, 128–bit long digital signatures. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 282–297. Springer, Heidelberg, April 2001.

[PCY+15] Albrecht Petzoldt, Ming-Shing Chen, Bo-Yin Yang, Chengdong Tao, and Jintai Ding. Design principles for HFEv-based multivariate signature schemes. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 311–334. Springer, Heidelberg, November / December 2015.

[PFH+20] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[PS20] Ray Perlner and Daniel Smith-Tone. Rainbow band separation is better than we thought. Cryptology ePrint Archive, Report 2020/702, 2020. https://eprint.iacr.org/2020/702.

[RGCB19] Prasanna Ravi, Sourav Sen Gupta, Anupam Chattopadhyay, and Shivam Bhasin. Improving speed of Dilithium's signing procedure. Cryptology ePrint Archive, Report 2019/420, 2019. https://eprint.iacr.org/2019/420.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[SK20] Kyung-Ah Shim and Namhun Koo. Algebraic fault analysis of UOV and Rainbow with the leakage of random vinegar values. *IEEE Trans. Inf. Forensics Secur.*, 15:2429–2439, 2020.

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

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[Tho13] Enrico Thomae. *About the security of multivariate quadratic public key schemes*. PhD thesis, Ruhr-UniversitÃđt Bochum, 2013.

[VBC+19] Javier A. Verbel, John Baena, Daniel Cabarcas, Ray A. Perlner, and Daniel Smith-Tone. On the complexity of "superdetermined" minrank instances. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 167–186. Springer, Heidelberg, 2019.

[WP05] Christopher Wolf and Bart Preneel. Large superfluous keys in multivariate quadratic asymmetric systems. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 275–287. Springer, Heidelberg, January 2005.

[WWCJ10] Jason M. Weaver, Kristin L. Wood, Richard H. Crawford, and Dan Jensen. Design of energy harvesting technology: Feasibility for low-power wireless sensor networks. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 44144, pages 463–472, 2010.

[ZBC19] Gustavo Zambonin, Matheus S. P. Bittencourt, and Ricardo Felipe Custódio. Handling vinegar variables to shorten Rainbow key pairs. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje eddine Rachidi, editors, *AFRICACRYPT 19*, volume 11627 of *LNCS*, pages 391–408. Springer, Heidelberg, July 2019.