

Name of Proposal:

QR-UOV

Principal Submitter:

Hiroki Furue

email: hiroki.furue@ntt.com

organization: NTT, Inc.

postal address: 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan

Auxiliary Submitters:

Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi,
Renma Sugai, Hiroshi Amagasa, Naofumi Homma,
Rei Ueno, Kosuke Sakata, Haruhisa Kosuge,
Kimihiro Yamakoshi, Koha Kinjo, Rika Akiyama,
Satoshi Nakamura, Shingo Orihara, Yuto Kawahara

Inventors: All listed submitters

Owners: All listed submitters

Backup Point of Contact:

Rika Akiyama

email: rika.akiyama@ntt.com

organization: NTT, Inc.

postal address: 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan

August 31, 2026

Contents

1	Introduction	7
1.1	History	7
1.2	Core Ideas of QR-UOV	7
1.3	Purpose and Scope	9
2	Notations and Parameters	10
3	Preliminaries	12
3.1	Basic Description of UOV used for QR-UOV	12
3.2	Matrix Representation of Quotient Ring Elements	13
4	Algorithm Specification	15
4.1	Key Generation	15
4.2	Signature Generation	18
4.3	Signature Verification	19
4.4	Representation of Keys and Signature	20
4.5	Parameter Sets	21
4.6	Auxiliary Functions	23
4.7	Generation of Pseudorandom Finite Field Elements	26
4.8	Pseudorandom Generator	28
4.9	Randomness in Signature Generation	28
4.10	Note on Basic Linear Algebra	29
5	Performance Analysis	30
6	Expected Security Strength	33
6.1	Underlying Problems and Security Definitions	33
6.2	Security Proof	34
6.3	Security Estimation of Proposed Parameters	38
7	Analysis of Attacks against QR-UOV	41
7.1	Claw Finding Attack	41
7.2	Direct Attack	41
7.3	Key Recovery Attacks on UOV	44
7.4	Irreducibility of Polynomial f	50
7.5	Lifting Method over Extension Field	51
7.6	Multiplication of Public Key Matrices by $(\Phi_x^f)^{(N)}$	54
8	Advantages and Limitations	57
A	Timing Attack Countermeasures	64
A.1	Overview	64
A.2	Security Analysis of Rank Information	64
A.3	Implementation Remarks	65
B	Additional Parameters	68
C	Expanded QR-UOV	70
C.1	Overview	70
C.2	Key Representation	70

C.3	Benchmark Results	71
D	QR-UOV with Full-Rank Signature Generation	72
E	Proofs of Lemmas for Security Reductions	73
E.1	Proof of Lemma 3	73
E.2	Proof of Lemma 5	79

Changelog (Round 2 $\rightarrow$ Round 3)

We made the following major changes to our submission:

- **Revised signature generation and verification algorithms:** To prevent an excessive increase in the signature-generation time, we restrict signature generation to cases where the coefficient matrix L of the linear system in the oil variables has rank at least $m - \Delta$. Here, m is the dimension of the square matrix L , and Δ is an additional parameter. When the rank of L is low, the expected number of retries in signature generation grows exponentially. We therefore choose Δ as small as possible, subject to the condition that, for a realistic upper bound on the number of signatures generated, the probability of encountering a matrix L with rank less than $m - \Delta$ remains below $1/2$ for the security proof. This choice balances the signature-generation time against the requirement that the rank threshold not affect the security proof.

In addition, to prevent a trivial forgery against the SUF-CMA (strong existential unforgeability against chosen-message attack) security that affects the previous versions, the algorithm compares the two signature candidates $\mathbf{s}$ and $-\mathbf{s}$ in lexicographic order and outputs the smaller one. Indeed, given a valid signature $\mathbf{s}$ on a queried message, an adversary can obtain another valid signature $-\mathbf{s}$ on the same message, thereby producing a fresh message-signature pair as a trivial forgery against SUF-CMA security. Accordingly, the verification algorithm checks that the received signature is the lexicographically smaller of $\mathbf{s}$ and $-\mathbf{s}$.

- **Parameters:** To provide a more refined selection of parameter sets suited to different implementation requirements, we provide two proposed parameter sets for each security level with $(q, \ell) = (127, 3)$ and $(q, \ell) = (127, 10)$. We retain the main parameter sets with $(q, \ell) = (127, 3)$ from the second-round specification and newly add those with $(q, \ell) = (127, 10)$. We select $q = 127$ for its implementation efficiency; the parameter sets with $\ell = 3$ are intended to provide balanced performance, whereas those with $\ell = 10$ offer relatively small public keys. The remaining parameter sets from the second-round specification are provided as additional parameters in the appendix.
- **Implementations:** Following the optimization techniques in [AUH26], we improve the QR-UOV implementations through algorithm-level optimizations and SIMD-based acceleration using AVX2, thereby improving overall performance.
- **Security analysis against key recovery attacks:** We added security analyses of QR-UOV against recent key-recovery attacks, including the wedge product attack [Ran26], the symmetric algebra attack [JPH⁺26], and the truncated polynomial ring attack [FI26], and confirmed that the

proposed parameter sets maintain the claimed security levels against these attacks.

- **Security analysis against timing attacks:** The NIST status report on the second-round candidates [NIS26] raised concerns about the cost of protecting QR-UOV signature generation against timing side-channel attacks. The signing procedure is not constant time specifically because the number of iterations required to sample a salt for which the linear system in the oil variables is solvable depends on the rank of the coefficient matrix L . Since making the signature-generation time independent of $\text{rank}(L)$ would incur substantial performance overhead, we do not attempt to conceal this timing dependence. Instead, we analyze the security impact of leaking this rank information, and our analysis shows no reduction in the estimated complexity of the known key-recovery attacks considered. We also consider an implementation strategy in which the remaining secret-dependent operations are executed in constant time, and evaluate and compare its performance.

Changelog (Round 1 $\rightarrow$ Round 2)

We made the following major changes to our submission:

- **Comprehensive algorithm specification and refined security proof:** In the first round, details on auxiliary functions such as seed expansion functions $\text{Expand}_{\text{pk}}$ and $\text{Expand}_{\text{sk}}$ were not specified. In this version, the specifications have been clarified to enhance the completeness of the algorithm specification. Since some auxiliary functions use a pseudorandom generator and rejection sampling, the security proof of QR-UOV has been refined to incorporate the use of these functions.
- **Implementations:** The NIST’s status report on the first round additional signatures [NIS24] mentioned that the implementation of QR-UOV can be further optimized. In response to this comment, we improve our implementation from the first round version. Notably, our new implementations can significantly reduce the timing data from the original implementation.
- **Security:** The status report to the first round candidates expresses their concerns about the security of the quotient ring (QR) structure of QR-UOV [NIS24]. However, our proposed parameter sets from the first round remain secure to this day, demonstrating the reliability of QR-UOV’s parameter selection. The security of QR-UOV is based on the hardness of UOV and QR-MQ problems. The UOV problem is a well-known problem, to which the security of other UOV variants is also reduced. We assume the UOV problem without the QR structure used in QR-UOV. The QR-MQ problem is an original assumption used in QR-UOV, and its hardness must be the primary concern raised in the status report; however, we sufficiently analyzed its security in the first round submission, and its security has not been weakened since the first round. Also, we add comments on some recently proposed attacks on UOV variants in the second round submission. In particular, we analyze the effect of the technique used in the attack on SNOVA [IA24] on QR-UOV.
- **Parameters:** In the first round submission, we propose four parameters for each security level. In this submission, we select one recommended parameter set for each level based on its performance as follows:

$$\text{I: } (q, v, m, \ell) = (127, 156, 54, 3),$$

$$\text{III: } (q, v, m, \ell) = (127, 228, 78, 3),$$

$$\text{V: } (q, v, m, \ell) = (127, 306, 105, 3).$$

1 Introduction

1.1 History

Currently used public key cryptosystems such as RSA and ECC can be broken in polynomial time using a quantum computer executing Shor’s algorithm [Sho99]. Thus, there has been growing interest in post-quantum cryptography (PQC), which is secure against quantum computing attacks. Indeed, the U.S. National Institute for Standards and Technology (NIST) has initiated a PQC standardization project [NIS].

Multivariate public key cryptography (MPKC), based on the difficulty of solving a system of multivariate quadratic polynomial equations over a finite field (the multivariate quadratic (MQ) problem), is regarded as a strong candidate for PQC. The MQ problem is NP-complete [GJ79] and is likely to be secure in the post-quantum era.

The unbalanced oil and vinegar signature scheme (UOV) [KPG99], a multivariate signature scheme proposed by Kipnis et al. at EUROCRYPT 1999, has withstood various types of attacks for over 20 years. UOV is a well-established signature scheme owing to its short signature and fast execution time. Indeed, a multilayer UOV variant Rainbow [DS05] was selected as a third-round finalist in the NIST PQC project [NIS20]. However, an attack on Rainbow proposed by Beullens in 2022 [Beu22] broke the security of third-round parameters; restoring security against this attack would make the scheme inefficient. This has induced a return of research focus from Rainbow to the original UOV. One drawback of UOV is that the public key is much larger than those of other PQC candidates, such as lattice-based signature schemes. Indeed, Rainbow, whose public key size is close to that of the plain UOV, had the largest public key among the third-round-finalist signature schemes, and NIST’s report [NIS20] stated that Rainbow is unsuitable as a general-purpose signature scheme owing to this problem.

One direction to solve the problem of UOV’s large public key size is to utilize an algebraic structure. CRYSTALS-Dilithium [DKL⁺18] is a selected algorithm in the NIST PQC project, and it is standardized as ML-DSA [FIP24]. It is based on the hardness of the module learning with errors ((M)LWE) problem [BGV14, LS15]. Non-structural LWE [Reg09] is a well-studied hard problem in cryptography, and the MLWE problem is its generalization using a module comprising vectors over a ring. This suggests that improving non-structural cryptographic schemes by investigating further algebraic theory is a natural direction to realize compact schemes. We present QR-UOV following this direction to develop a UOV variant with a small public key.

1.2 Core Ideas of QR-UOV

The core idea of QR-UOV is to impose quotient-ring structure on the blocks of the UOV public-key matrices, thereby reducing the number of field elements required to represent each block. The public key of QR-UOV is represented

by block matrices in which every component corresponds to an element of a quotient ring. More precisely, we use an injective ring homomorphism from the quotient ring $\mathbb{F}_q[x]/(f)$ to the matrix ring $\mathbb{F}_q^{\ell \times \ell}$, where $f \in \mathbb{F}_q[x]$ is a polynomial with $\deg f = \ell$. The image Φ_g^f of the homomorphism for $g \in \mathbb{F}_q[x]/(f)$ is called the *polynomial matrix* of g . From this homomorphism, we can compress the ℓ^2 components in Φ_g^f to ℓ elements of $\mathbb{F}_q$ because the polynomial matrix Φ_g^f is determined by the ℓ coefficients of g . This can be considered as a generalization of the block-anti-circulant UOV (BAC-UOV) presented at SAC 2019 [SP20], which is the case for $f = x^\ell - 1$. Utilizing the elements of a quotient ring in block matrices is similar to the MLWE problem [BGV14, LS15] because the MLWE problem uses elements of a ring in vectors. Namely, the transition from UOV to QR-UOV (including BAC-UOV) can be regarded as analogous to the transition from LWE to MLWE. Therefore, as with the MLWE problem, this type of research deserves more attention than passing notice.

QR-UOV requires the public key to be generated considering the symmetry of the polynomial matrices Φ_g^f . In UOV, the public key $\mathcal{P} = (p_1, \dots, p_m)$, which comprises quadratic polynomials p_i , is obtained by composing a central map $\mathcal{F} = (f_1, \dots, f_m)$ and a linear map $\mathcal{S}$, that is, $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$. Then, the corresponding matrices $P_1, \dots, P_m$ of the public key $\mathcal{P}$ are given by $P_i = S^\top F_i S$, where $F_1, \dots, F_m$, and S are matrices corresponding to $\mathcal{F}$ and $\mathcal{S}$, respectively. We cannot construct $P_1, \dots, P_m$ as block matrices whose components are polynomial matrices Φ_g^f by naively choosing $F_1, \dots, F_m$ and S as block matrices of Φ_g^f . This is because polynomial matrices Φ_g^f are generally unstable under the transpose operation. To solve this problem, we introduce the concept of an $\ell \times \ell$ invertible matrix W such that $W\Phi_g^f$ is symmetric for any $g \in \mathbb{F}_q[x]/(f)$; that is, $W\Phi_g^f$ is stable under the transpose operation. Furue et al. [FIKT21] showed that there exists such a symmetric W for any quotient ring $\mathbb{F}_q[x]/(f)$. Then, from the equations

$$(\Phi_{g_1}^f)^\top (W\Phi_{g_2}^f)\Phi_{g_1}^f = (W\Phi_{g_1}^f)^\top \Phi_{g_2}^f \Phi_{g_1}^f = W\Phi_{g_1 g_2 g_1}^f,$$

we can embed the $W\Phi_g^f$ structure into public key matrices by choosing $F_1, \dots, F_m$ as block matrices with $W\Phi_g^f$ and S as a block matrix with Φ_g^f .

Moreover, we should consider how the choice of f affects the security of QR-UOV. Indeed, Furue et al. [FKI⁺20] broke BAC-UOV by transforming its anti-circulant matrices into diagonal concatenations of two smaller matrices. This transformation is obtained from the decomposition $x^\ell - 1 = (x - 1)(x^{\ell-1} + \dots + 1)$. Therefore, we investigate the relationship between the irreducibility of the polynomial f used to generate the quotient ring $\mathbb{F}_q[x]/(f)$ and the existence of such a transformation for symmetric matrices $W\Phi_g^f$. As shown in [FIKT21], if f is irreducible (i.e., $\mathbb{F}_q[x]/(f)$ is a field), then there is no such transformation for matrices $W\Phi_g^f$, indicating that such an f is resistant to Furue et al.'s structural attack [FKI⁺20].

1.3 Purpose and Scope

In this document, we present a multivariate polynomial based digital signature scheme QR-UOV. The purpose and scope of this document are as follows:

- We specify the QR-UOV algorithms and parameter sets targeting NIST security levels I, III, and V.
- We establish a security reduction to the underlying hardness assumptions in the quantum random oracle model (QROM).
- We provide a comprehensive concrete security analysis of the underlying hardness assumptions against state-of-the-art cryptanalytic techniques.
- We provide reference and optimized implementations, and evaluate their performance.

Organization The rest of this document is organized as follows. Section 2 prepares some notations. Section 3 gives preliminaries on describing QR-UOV. Section 4 describes the algorithm specification of QR-UOV. Section 5 provides the performance analysis. Section 6 gives our security statements. Section 7 analyzes concrete attacks on QR-UOV. Section 8 discusses the advantages and limitations of QR-UOV.

Acknowledgements We are grateful for help from Makoto Yanagisawa and Atsuhito Nakase. We also acknowledge Noriki Mo for pointing out inconsistencies between the spec and implementation. We are grateful to Shuhei Nakamura for his useful technical comments. We thank Masahiro Mizukami, Sari Handa, Maki Adachi, Tomoyuki Okazaki, Ayumi Ito, and Tetsutaro Kobayashi for implementing timing-attack-resilient prototypes. We also thank Masashi Mizukane, Takashi Yamashita, and Shin-ichiro Nishiyama for improving the Arm Cortex-M4 implementations.

2 Notations and Parameters

This section describes the notations and parameters used in this document. We also define more specific basic methods used later in the document.

bit	one of the two symbols ‘0’ or ‘1’.
bit string	an ordered sequence of bits.
byte	a bit string of length 8.
byte string	an ordered sequence of bytes.
$(\mathbf{a} \parallel \mathbf{b})$	the concatenation of two given vectors $\mathbf{a}$ and $\mathbf{b}$.
$\mathbf{0}_a$	the a -dimensional zero vector over finite fields.
$\mathbb{F}_q$	finite field with q elements for a prime power q .
$\mathbb{B}$	the set $\{0, \dots, 255\}$ of integers represented by a byte.
$\lceil x \rceil$	the smallest integer greater than or equal to a given real number x .
$\lfloor x \rfloor$	the largest integer less than or equal to a given real number x .
$[n]$	the set $\{1, \dots, n\}$ for a given positive integer n .
$a \xleftarrow{\$} A$	$a \in A$ is chosen uniformly at random from A .
$\text{IntegerToBits}(x, \alpha)$	computes a base-2 representation of $x \bmod 2^\alpha$ using big-endian order, ensuring the output is an α -bit string.
$\text{IntegerToBytes}(x, \alpha)$	computes a base-256 representation of $x \bmod 256^\alpha$ using big-endian order, ensuring the output is an α -byte string.
$\text{BitsToInteger}(x, \alpha)$	computes an integer value from the α -bit string x using big-endian order.
$\text{BytesToInteger}(r)$	computes an integer value from $r \in \mathbb{B}$.
$\text{BytesToBits}(x)$	converts a byte string x to a bit string using big-endian order.
$\text{Trunc}_l(x)$	returns the leftmost l bytes of a byte string x .

We introduce the parameters used in the specification of QR-UOV.

q	number of elements of the finite field.
ℓ	degree of the polynomial defining the quotient ring.
v	number of vinegar variables, set to be a multiple of ℓ .
m	number of oil variables (equals the number of equations), set to be a multiple of ℓ .
n	number of variables: $n = v + m$.
V	$V = v/\ell$.
M	$M = m/\ell$.
N	$N = V + M$.
λ	security parameter.
sk_seed_len	length of private key seed (in bits).
pk_seed_len	length of public key seed (in bits).
τ	length of random byte string input for rejection sampling, which takes three possible values depending on its specific usage.
Δ	nonnegative integer specifying the minimum acceptable rank $m - \Delta$ of the coefficient matrix L in signature generation.

Here, we also give some notations for representation matrices of elements of a quotient ring described in Subsection 3.2.

f	an irreducible polynomial in $\mathbb{F}_q[x]$ with degree ℓ .
Φ_g^f	an $\ell \times \ell$ matrix over $\mathbb{F}_q$ corresponding to $g \in \mathbb{F}_q[x]/(f)$ defined by equation (4) in Subsection 3.2.
$\mathcal{A}_f$	a set $\{\Phi_g^f \in \mathbb{F}_q^{\ell \times \ell} \mid g \in \mathbb{F}_q[x]/(f)\}$.
W	an $\ell \times \ell$ matrix over $\mathbb{F}_q$ such that WX is symmetric for any $X \in \mathcal{A}_f$.
$W\mathcal{A}_f$	a set $\{WX \in \mathbb{F}_q^{\ell \times \ell} \mid X \in \mathcal{A}_f\}$.
$\mathcal{A}_f^{a,b}$	the set of $a\ell \times b\ell$ block matrices whose blocks are elements of $\mathcal{A}_f$.
$W^{(a)}$	the $a\ell \times a\ell$ block diagonal matrix concatenating W diagonally a times.

3 Preliminaries

Since QR-UOV is an extension of plain UOV, we first review the construction of plain UOV in Subsection 3.1 to facilitate a description of QR-UOV in Section 4. Furthermore, as a preliminary step for the construction of QR-UOV, we introduce matrices representing elements of a quotient ring in Subsection 3.2.

3.1 Basic Description of UOV used for QR-UOV

This subsection describes the structure of the unbalanced oil and vinegar signature scheme (UOV) [KPG99]. For variables $\mathbf{x} = (x_1, \dots, x_n)$ over $\mathbb{F}_q$, we call $x_1, \dots, x_v$ *vinegar variables* and $x_{v+1}, \dots, x_n$ *oil variables*.

We first recall the key generation of UOV as follows: We design $\mathcal{F} = (f_1, \dots, f_m) : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$, called a *central map*, such that each f_k with $k \in [m]$ is a quadratic polynomial of the form

$$f_k(x_1, \dots, x_n) = \sum_{i=1}^v \sum_{j=i}^n \alpha_{i,j}^{(k)} x_i x_j, \quad (1)$$

where $\alpha_{i,j}^{(k)} \in \mathbb{F}_q$. Next, we choose a random invertible linear map $\mathcal{S} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ to hide the structure of $\mathcal{F}$. The public key map $\mathcal{P}$ is then provided as a polynomial map,

$$\mathcal{P} = \mathcal{F} \circ \mathcal{S} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m, \quad (2)$$

and the private key comprises $\mathcal{F}$ and $\mathcal{S}$. Here, we omit linear and constant terms of $\mathcal{F}$ and constant terms of $\mathcal{S}$ for simplicity.

Next, we describe the inversion of the central map $\mathcal{F}$. When we try to find $\mathbf{x} \in \mathbb{F}_q^n$ satisfying $\mathcal{F}(\mathbf{x}) = \mathbf{t}$ for a given $\mathbf{t} \in \mathbb{F}_q^m$, we first choose random values $y_1, \dots, y_v$ in $\mathbb{F}_q$ as the values of the vinegar variables. We can then easily obtain a solution for the equation $\mathcal{F}(y_1, \dots, y_v, x_{v+1}, \dots, x_n) = \mathbf{t}$, because this is a linear system of m equations in $n - v$ oil variables from the construction of the central map (1). If there is no solution to this equation, we choose new random values $y'_1, \dots, y'_v$, and repeat the above procedure.

By using this inversion approach, the signature is generated as follows: Given a message $\mathbf{t} \in \mathbb{F}_q^m$ to be signed, find a solution $\mathbf{y}$ to the equation $\mathcal{F}(\mathbf{x}) = \mathbf{t}$, and this gives a signature $\mathbf{s} = \mathcal{S}^{-1}(\mathbf{y}) \in \mathbb{F}_q^n$ for the message $\mathbf{t}$. The verification is performed by confirming whether $\mathcal{P}(\mathbf{s}) = \mathbf{t}$.

Finally, we introduce matrices representing the public and private keys of UOV. For each polynomial p_i of the public key map $\mathcal{P}$, there exists an $n \times n$ matrix P_i such that $p_i(\mathbf{x}) = \mathbf{x}^\top \cdot P_i \cdot \mathbf{x}$. We call the matrices $P_1, \dots, P_m$ the public key matrices. Similarly, an $n \times n$ matrix F_i can be taken for each f_i with $i \in [m]$, and an $n \times n$ matrix S is defined to satisfy $\mathcal{S}(\mathbf{x}) = S \cdot \mathbf{x}$. In general, these matrices P_i and F_i are taken as symmetric matrices if q is odd, and are taken as upper triangular matrices if q is even. For these representation matrices, based on equation (1), F_i has the following form

$$\begin{pmatrix} *_{v \times v} & *_{v \times m} \\ *_{m \times v} & 0_{m \times m} \end{pmatrix}. \quad (3)$$

Furthermore, from $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$, we have

$$P_i = S^\top F_i S \quad (i \in [m]).$$

3.2 Matrix Representation of Quotient Ring Elements

We introduce polynomial matrices representing elements of a quotient ring. These matrices are utilized for the construction of QR-UOV in Section 4.

Let ℓ be a positive integer and $f \in \mathbb{F}_q[x]$ with $\deg f = \ell$. For any element g of the quotient ring $\mathbb{F}_q[x]/(f)$, we can uniquely define an $\ell \times \ell$ matrix Φ_g^f over $\mathbb{F}_q$ such that

$$\begin{pmatrix} 1 & x & \cdots & x^{\ell-1} \end{pmatrix} \Phi_g^f = \begin{pmatrix} g & xg & \cdots & x^{\ell-1}g \end{pmatrix}. \quad (4)$$

From this equation, we have

$$x^{j-1}g = \sum_{i=1}^{\ell} (\Phi_g^f)_{ij} \cdot x^{i-1} \quad (j \in [\ell]),$$

and $(\Phi_g^f)_{ij}$ is the coefficient of x^{i-1} in $x^{j-1}g$. We call the matrix Φ_g^f the *polynomial matrix* of g . The following lemma can be easily derived from this definition:

Lemma 1. *For any $g_1, g_2 \in \mathbb{F}_q[x]/(f)$, we have*

$$\Phi_{g_1}^f + \Phi_{g_2}^f = \Phi_{g_1+g_2}^f, \quad \Phi_{g_1}^f \Phi_{g_2}^f = \Phi_{g_1 g_2}^f.$$

That is, the map $g \mapsto \Phi_g^f$ is an injective ring homomorphism from $\mathbb{F}_q[x]/(f)$ to the matrix ring $\mathbb{F}_q^{\ell \times \ell}$.

We let the algebra of the matrices $\mathcal{A}_f := \{\Phi_g^f \in \mathbb{F}_q^{\ell \times \ell} \mid g \in \mathbb{F}_q[x]/(f)\}$. This is a subalgebra in the matrix algebra $\mathbb{F}_q^{\ell \times \ell}$ from Lemma 1. Every $\ell \times \ell$ polynomial matrix Φ_g^f in $\mathcal{A}_f$ can be represented by only ℓ elements in $\mathbb{F}_q$, because Φ_g^f is determined by the ℓ coefficients of $g \in \mathbb{F}_q[x]/(f)$. QR-UOV compresses the public key size of UOV by utilizing this property of Φ_g^f .

For the construction of QR-UOV, we also introduce the concept of a matrix $W \in \mathbb{F}_q^{\ell \times \ell}$ such that $W\Phi_g^f$ is stable under the transpose operation. Note that any matrix in $W\mathcal{A}_f := \{WX \in \mathbb{F}_q^{\ell \times \ell} \mid X \in \mathcal{A}_f\}$ can also be represented by only ℓ elements in $\mathbb{F}_q$. The following theorem shows that there exists an invertible matrix W for any f .

Theorem 1 (Theorem 1 in [FIKT21]). *Let $f \in \mathbb{F}_q[x]$ with $\deg f = \ell$. Then, there exists an invertible matrix $W \in \mathbb{F}_q^{\ell \times \ell}$ such that WX is a symmetric matrix for any $X \in \mathcal{A}_f$.*

In the proof of Theorem 1 in [FIKT21], the authors propose a way of constructing W satisfying the condition from a nonzero linear map $\bar{\phi} : \mathbb{F}_q[x]/(f) \rightarrow \mathbb{F}_q$ such that the ij -component of W is equal to $\bar{\phi}(x^{i+j-2})$.

For $\mathcal{A}_f$ and positive integers N , we define the set $\mathcal{A}_f^{N,N}$ of block matrices in $\mathbb{F}_q^{\ell N \times \ell N}$ whose blocks are elements of $\mathcal{A}_f$.

Example 1. For $f = x^3 - 3x - 1$ in $\mathbb{F}_7[x]$, we can take one element of $\mathcal{A}_f^{2;2}$ as follows

$$\begin{pmatrix} 2 & 0 & 2 & 0 & 0 & 1 \\ 2 & 2 & 6 & 1 & 0 & 3 \\ 0 & 2 & 2 & 0 & 1 & 0 \\ 3 & 2 & 5 & 3 & 6 & 5 \\ 5 & 2 & 3 & 5 & 0 & 0 \\ 2 & 5 & 2 & 6 & 5 & 0 \end{pmatrix}.$$

Every 3×3 block of this matrix can be represented as an element of $\mathbb{F}_q[x]/(f)$, that is, this matrix can be represented as a 2×2 matrix over $\mathbb{F}_q[x]/(f)$

$$\begin{pmatrix} 2 + 2x & x \\ 3 + 5x + 2x^2 & 3 + 5x + 6x^2 \end{pmatrix}.$$

We construct QR-UOV using $\mathcal{A}_f$ with an irreducible f and the extension field $\mathbb{F}_{q^\ell} \simeq \mathbb{F}_q[x]/(f)$. See Subsection 7.4 for the rationale behind using an irreducible polynomial f in QR-UOV.

4 Algorithm Specification

This section defines the algorithms for functions composing QR-UOV. First, we present the algorithms of key generation, signature generation, and signature verification in Subsections 4.1, 4.2, and 4.3. Subsection 4.4 specifies the representations of the public key, private key, and signature, and Subsection 4.5 presents the proposed parameter sets. Subsection 4.6 defines the auxiliary functions used in the main algorithms, while Subsections 4.7 and 4.8 describe the generation of pseudorandom finite field elements and the pseudorandom generator, respectively. Subsection 4.9 specifies the randomness used in signature generation. Finally, Subsection 4.10 provides remarks on the basic linear-algebra operations required by the scheme.

4.1 Key Generation

The algorithm **KeyGen** (Algorithm 1) outputs a public key $\mathbf{pk}$ and a private key $\mathbf{sk}$. We first recall some notations and give a naive way of generating the public and private keys of QR-UOV. We then provide a strategy to reduce the public/private key sizes by restricting the private key to a compact form and storing expandable data as a seed within the keys. It should be noted that not all data of the public key can be stored as a seed. Finally, we explain a method of representing the extended public and private keys by matrices over the extension field $\mathbb{F}_{q^\ell}$.

Algorithm 1 **KeyGen**()

Output: public key $\mathbf{pk} \in \{0,1\}^{\mathbf{pk_seed_len}} \times \left(\mathbb{F}_{q^\ell}^{M \times M}\right)^m$ and private key $\mathbf{sk} \in \{0,1\}^{\mathbf{sk_seed_len}}$

- 1: $\mathbf{seed_sk} \xleftarrow{\$} \{0,1\}^{\mathbf{sk_seed_len}}$
- 2: $(\bar{S}', \mathbf{seed_pk}) \leftarrow \text{Expand}_{\mathbf{sk}}(\mathbf{seed_sk}) \quad \triangleright \bar{S}' \in \mathbb{F}_{q^\ell}^{V \times M}$
- 3: **for** i from 1 to m **do**
- 4: $(\bar{P}_{i,1}, \bar{P}_{i,2}) \leftarrow \text{Expand}_{\mathbf{pk}}(\mathbf{seed_pk}, i) \quad \triangleright \bar{P}_{i,1} \in \mathbb{F}_{q^\ell}^{V \times V} \text{ (symmetric)}, \bar{P}_{i,2} \in \mathbb{F}_{q^\ell}^{V \times M}$
- 5: $\bar{P}_{i,3} \leftarrow -\bar{S}'^\top \bar{P}_{i,1} \bar{S}' + \bar{P}_{i,2}^\top \bar{S}' + \bar{S}'^\top \bar{P}_{i,2} \quad \triangleright \bar{P}_{i,3} \in \mathbb{F}_{q^\ell}^{M \times M}$
- 6: **end for**
- 7: **return** $(\mathbf{pk}, \mathbf{sk}) = ((\mathbf{seed_pk}, \{\bar{P}_{i,3}\}_{i \in [m]}), \mathbf{seed_sk})$

We begin with the notation and naive key generation of QR-UOV. As mentioned in Section 2, let v be the number of vinegar variables, m be the number of oil variables which is equal to the number of quadratic polynomials, and $n = v + m$. We set v , m , and n as multiples of the parameter ℓ , namely $v = \ell \cdot V$, $m = \ell \cdot M$, and $n = \ell \cdot N$. The public and private keys of QR-UOV are represented by elements of $\mathcal{A}_f^{N,N}$ and $W^{(N)} \mathcal{A}_f^{N,N} := \left\{ W^{(N)} \cdot X \mid X \in \mathcal{A}_f^{N,N} \right\}$ (see the definitions in Section 3.2). Note that we use an irreducible polynomial

as f of $\mathcal{A}_f$ from a security perspective stated in Subsection 7.4.

We give a naive way of generating the public and private keys of QR-UOV.

1. Choose F_i ($i \in [m]$) from $W^{(N)}\mathcal{A}_f^{N,N}$ as a symmetric matrix with the lower-right $m \times m$ zero-block as in (3).
2. Choose an invertible matrix S from $\mathcal{A}_f^{N,N}$ randomly.
3. Compute the public key matrices $P_i = S^\top F_i S$ ($i \in [m]$).

Then, P_i ($i \in [m]$) representing the public key map are elements of $W^{(N)}\mathcal{A}_f^{N,N}$ from the following proposition:

Proposition 1 (Proposition 1 in [FIKT21]). *For $X \in \mathcal{A}_f^{N,N}$ and $Y \in W^{(N)}\mathcal{A}_f^{N,N}$, we have*

$$X^\top Y X \in W^{(N)}\mathcal{A}_f^{N,N}.$$

Next, we present the key generation adopted in this specification, which can reduce the public/private key sizes. We apply a method restricting S to a specific compact form, which was first proposed by Czypek et al. [CHT12]. For P_i ($i \in [m]$) and F_i ($i \in [m]$), we define submatrices as follows:

$$P_i = \begin{pmatrix} P_{i,1} & P_{i,2} \\ P_{i,2}^\top & P_{i,3} \end{pmatrix},$$

$$F_i = \begin{pmatrix} F_{i,1} & F_{i,2} \\ F_{i,2}^\top & 0_{m \times m} \end{pmatrix},$$

where $P_{i,1}$ and $F_{i,1}$ are symmetric $v \times v$ matrices, $P_{i,2}$ and $F_{i,2}$ are $v \times m$ matrices, and $P_{i,3}$ is a symmetric $m \times m$ matrix. We then restrict S to the following compact form

$$S = \begin{pmatrix} I_v & S' \\ 0_{m \times v} & I_m \end{pmatrix}, \quad (5)$$

where S' is a $v \times m$ matrix. Then, from $P_i = S^\top F_i S$ ($i \in [m]$), we obtain

$$\begin{aligned} F_{i,1} &= P_{i,1}, \\ F_{i,2} &= -P_{i,1}S' + P_{i,2}, \\ 0_{m \times m} &= S'^\top P_{i,1}S' - P_{i,2}^\top S' - S'^\top P_{i,2} + P_{i,3}. \end{aligned} \quad (6)$$

Therefore, we can compute

$$P_{i,3} = -S'^\top P_{i,1}S' + P_{i,2}^\top S' + S'^\top P_{i,2}.$$

Then, we can take $P_{i,1}$ and $P_{i,2}$ by expanding seeds and compress the public key into $m \times m$ matrices $P_{i,3}$ ($i \in [m]$) and the public seed seed_{pk} of length pk.seed_len bits for $P_{i,1}$, $P_{i,2}$ ($i \in [m]$). Similarly, we compress the private key into the private seed seed_{sk} of length sk.seed_len bits for S' . Algorithm 1 uses two functions $\text{Expand}_{\text{sk}}$ and $\text{Expand}_{\text{pk}}$ given in Subsection 4.6 to expand the public and private keys from the seeds.

Finally, we explain the method of transforming matrices into the extension field $\mathbb{F}_{q^\ell}$ since Algorithm 1 computes matrix operations over $\mathbb{F}_{q^\ell}$ for efficiency. This method is called *the pull-back method* [FIKT21]. Since the public key matrices P_k with $k \in [m]$ are elements of $W^{(N)}\mathcal{A}_f^{N,N}$, we can take ℓ matrices $\bar{P}_k^{(0)}, \dots, \bar{P}_k^{(\ell-1)} \in \mathbb{F}_q^{N \times N}$ satisfying

$$P_k = \sum_{i=0}^{\ell-1} \left(\bar{P}_k^{(i)} \otimes W\Phi_{x^i}^f \right),$$

where $\otimes$ denotes the tensor product and, for $A = (a_{i,j})_{i,j} \in \mathbb{F}_q^{m \times n}$ and $B = (b_{i,j})_{i,j} \in \mathbb{F}_q^{r \times s}$, we have

$$A \otimes B = \begin{pmatrix} a_{1,1}B & \cdots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \cdots & a_{m,n}B \end{pmatrix} \in \mathbb{F}_q^{mr \times ns}.$$

We then can define an $N \times N$ matrix $\bar{P}_k$ over $\mathbb{F}_q[x]/(f) \cong \mathbb{F}_{q^\ell}$ as follows:

$$\bar{P}_k = \sum_{i=0}^{\ell-1} x^i \bar{P}_k^{(i)}.$$

In the same way, we can construct $\bar{F}_1, \dots, \bar{F}_m$ and $\bar{S}$ over $\mathbb{F}_{q^\ell}$ corresponding to $F_1, \dots, F_m$ and S as follows:

$$\begin{aligned} F_k &= \sum_{i=0}^{\ell-1} \left(\bar{F}_k^{(i)} \otimes W\Phi_{x^i}^f \right) \Rightarrow \bar{F}_k = \sum_{i=0}^{\ell-1} x^i \bar{F}_k^{(i)}, \\ S &= \sum_{i=0}^{\ell-1} \left(\bar{S}^{(i)} \otimes \Phi_{x^i}^f \right) \Rightarrow \bar{S} = \sum_{i=0}^{\ell-1} x^i \bar{S}^{(i)}. \end{aligned}$$

It follows from $P_k = S^\top F_k S$ that $\bar{P}_k = \bar{S}^\top \bar{F}_k \bar{S}$, and $\bar{F}_k$ has the form as in (3). Furthermore, we define a function ϕ corresponding to the transformation from matrices over $\mathbb{F}_{q^\ell}$ to block matrices of elements in $\mathcal{A}_f$ as follows:

$$\phi : \mathbb{F}_{q^\ell}^{a \times b} \ni \begin{pmatrix} g_{1,1} & \cdots & g_{1,b} \\ \vdots & \ddots & \vdots \\ g_{a,1} & \cdots & g_{a,b} \end{pmatrix} \mapsto \begin{pmatrix} \Phi_{g_{1,1}}^f & \cdots & \Phi_{g_{1,b}}^f \\ \vdots & \ddots & \vdots \\ \Phi_{g_{a,1}}^f & \cdots & \Phi_{g_{a,b}}^f \end{pmatrix} \in \mathcal{A}_f^{a,b}.$$

For each $k \in [m]$, we then have

$$\phi(\bar{P}_k) = \left(W^{(N)} \right)^{-1} P_k, \quad \phi(\bar{F}_k) = \left(W^{(N)} \right)^{-1} F_k, \quad \phi(\bar{S}) = S.$$

4.2 Signature Generation

The algorithm **Sign** (Algorithm 2) takes a message $\mathbf{M}$ and a private key $\mathbf{sk}$ as input. A single invocation either outputs a signature $\sigma \in \{0, 1\}^\lambda \times \mathbb{F}_q^n$ or returns the distinguished failure symbol $\perp$. The procedure primarily follows the standard signature generation of the plain UOV: Invert the central map $\mathcal{F}$ by fixing v values of the vinegar variables, and then multiply S^{-1} in the form of

$$S^{-1} = \begin{pmatrix} I_v & -S' \\ 0_{m \times v} & I_m \end{pmatrix},$$

from equation (5). We add a modification for the EUF-CMA (existential unforgeability against chosen-message attack) security proof proposed by Sakumoto et al. [SSH11].

Algorithm 2 **Sign**($\mathbf{M}, \mathbf{sk}$)

Input: message $\mathbf{M} \in \mathbb{B}^*$ and private key $\mathbf{sk} \in \{0, 1\}^{\mathbf{sk_seed_len}}$

Output: signature $\sigma \in (\{0, 1\}^\lambda \times \mathbb{F}_q^n) \cup \{\perp\}$

```

1:  $\mathbf{seed}_{\mathbf{sk}} \leftarrow \mathbf{sk}$ 
2:  $(\bar{S}', \mathbf{seed}_{\mathbf{pk}}) \leftarrow \text{Expand}_{\mathbf{sk}}(\mathbf{seed}_{\mathbf{sk}})$ 
3:  $\mathbf{y} = (y_1, \dots, y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$ 
4: for  $i$  from 1 to  $m$  do
5:    $(\bar{P}_{i,1}, \bar{P}_{i,2}) \leftarrow \text{Expand}_{\mathbf{pk}}(\mathbf{seed}_{\mathbf{pk}}, i)$ 
6:    $\mathbf{L}_i \leftarrow -2\phi(\bar{S}')^\top W^{(V)}\phi(\bar{P}_{i,1})\mathbf{y} + 2W^{(M)}\phi(\bar{P}_{i,2})\mathbf{y}$   $\triangleright \mathbf{L}_i \in \mathbb{F}_q^m$ 
7:    $u_i \leftarrow \mathbf{y}^\top W^{(V)}\phi(\bar{P}_{i,1})\mathbf{y}$   $\triangleright u_i \in \mathbb{F}_q$ 
8: end for
9:  $L \leftarrow (\mathbf{L}_1, \dots, \mathbf{L}_m)^\top$   $\triangleright L \in \mathbb{F}_q^{m \times m}$ 
10: if  $\text{rank}(L) < m - \Delta$  then
11:   return  $\perp$ 
12: end if
13:  $\mathbf{u} \leftarrow (u_1, \dots, u_m)^\top$   $\triangleright \mathbf{u} \in \mathbb{F}_q^m$ 
14:  $\mu \leftarrow \text{SHAKE256}(\mathbf{seed}_{\mathbf{pk}} \parallel \text{BytesToBits}(\mathbf{M}), 512)$ 
15: repeat
16:    $\rho \xleftarrow{\$} \{0, 1\}^\lambda$ 
17:    $\mathbf{t} \leftarrow \text{Hash}(\mu, \rho)$   $\triangleright \mathbf{t} \in \mathbb{F}_q^m$ 
18: until  $L\mathbf{x} = \mathbf{t} - \mathbf{u}$  has solutions for  $\mathbf{x}$ .
19: Choose one solution  $(y_{v+1}, \dots, y_n)^\top \in \mathbb{F}_q^{m-n}$  of  $L\mathbf{x} = \mathbf{t} - \mathbf{u}$  randomly.
20:  $\mathbf{s} \leftarrow (y_1, \dots, y_v, y_{v+1}, \dots, y_n)^\top - (\phi(\bar{S}') \cdot (y_{v+1}, \dots, y_n)^\top \parallel \mathbf{0}_m)$   $\triangleright \mathbf{s} \in \mathbb{F}_q^n$ 
21: if  $-\mathbf{s}$  is lexicographically smaller than  $\mathbf{s}$  then
22:    $\mathbf{s} \leftarrow -\mathbf{s}$ 
23: end if
24: return  $\sigma = (\rho, \mathbf{s})$ 
```

We describe the inversion of the central map $\mathcal{F}$ in the signature generation

of QR-UOV. We first choose values for the vinegar variables $y_1, \dots, y_v \in \mathbb{F}_q$ randomly. We then choose λ -bit random salt ρ and compute $\mathbf{t} \in \mathbb{F}_q^m$ by applying Hash on the input concatenating a bit string μ and the salt ρ , namely $\mathbf{t} := \text{Hash}(\mu, \rho)$, where μ is a hash value of the public seed seed_{pk} and the message $\mathbf{M}$. If the linear system in the oil variables $x_{v+1}, \dots, x_n$,

$$\mathcal{F}(y_1, \dots, y_v, x_{v+1}, \dots, x_n) = \mathbf{t}, \quad (7)$$

has at least one solution, then we obtain the signature by applying $\mathcal{S}^{-1}$ to $(y_1, \dots, y_v, y_{v+1}, \dots, y_n)$, where $(y_{v+1}, \dots, y_n)$ is a randomly chosen solution of equation (7). If equation (7) has no solution, then we choose a new salt ρ and update $\mathbf{t}$ until equation (7) has at least one solution. Note that Algorithm 2 performs some operation over extension fields as in Algorithm 1 and uses the function ϕ defined in Subsection 4.1. In particular, in line 6, $\mathbf{L}_i$, which corresponds to the i -th row of a matrix representing the linear equation (7), is computed as follows:

$$\begin{aligned} \mathbf{L}_i &= (-2S'^\top P_{i,1} + 2P_{i,2}^\top) \mathbf{y} \\ &= -2\phi(\bar{S}')^\top W^{(V)} \phi(\bar{P}_{i,1}) \mathbf{y} + 2W^{(M)} \phi(\bar{P}_{i,2}^\top) \mathbf{y}, \end{aligned}$$

since $(W^{(V)} \phi(\bar{P}_{i,2}))^\top = W^{(M)} \phi(\bar{P}_{i,2}^\top)$ due to the symmetry of $W\Phi_g^f$.

The main difference from the signature generation of the plain UOV is that if equation (7) has no solution, then it chooses a new random salt instead of selecting new values for the vinegar variables. By doing so, when the rank restriction and canonicalization are omitted, the signature $\mathbf{s}$ satisfying $\mathcal{P}(\mathbf{s}) = \text{Hash}(\mu, \rho)$ is uniformly distributed over $\mathbb{F}_q^n$, and this fact enables us to prove the EUF-CMA security of the underlying signing procedure. We show in Subsection 6.2 that the proof remains valid even after introducing both the rank restriction and canonicalization.

We make two further modifications to the signature-generation procedure. First, we restrict the coefficient matrix L to satisfy $\text{rank}(L) \geq m - \Delta$. This restriction keeps the number of loop iterations required for signature generation relatively small, except with negligible probability, and thus prevents the loop from becoming a performance bottleneck.

Second, we canonicalize the signature by comparing the two signature candidates $\mathbf{s}$ and $-\mathbf{s}$ in lexicographic order and outputting the smaller one. For two distinct vectors $\mathbf{a} = (a_1, \dots, a_n)^\top$ and $\mathbf{b} = (b_1, \dots, b_n)^\top$ in $\mathbb{F}_q^n$, we say that $\mathbf{a}$ is lexicographically smaller than $\mathbf{b}$ if $a_j < b_j$ for the smallest index j such that $a_j \neq b_j$, where the elements of $\mathbb{F}_q$ are identified with the integers in $\{0, \dots, q-1\}$. This canonicalization prevents an adversary from deriving the distinct signature $-\mathbf{s}$ from a valid signature $\mathbf{s}$ for the same message, thereby preventing this trivial violation of SUF-CMA security.

4.3 Signature Verification

The algorithm Verify (Algorithm 3) takes a message $\mathbf{M}$, a public key pk , and a signature σ as input and outputs accept or reject. The procedure follows that

of the plain UOV. For an overview, the signature is verified as:

- Compute $\mathbf{t} = \text{Hash}(\mu, \rho)$, where μ is a hash value of seed_{pk} and $\mathbf{M}$.
- Compute $\mathbf{t}' \in \mathbb{F}_q^m$ by substituting the signature $\mathbf{s} \in \mathbb{F}_q^n$ into the public key map $\mathcal{P}$, namely $\mathbf{t}' = \mathcal{P}(\mathbf{s})$.
- If $\mathbf{t} = \mathbf{t}'$ holds, the signature σ is accepted, otherwise it is rejected.

Algorithm 3 $\text{Verify}(\mathbf{M}, \text{pk}, \sigma)$

Input: message $\mathbf{M} \in \mathbb{B}^*$, public key $\text{pk} \in \{0, 1\}^{\text{pk_seed_len}} \times \left(\mathbb{F}_{q^\ell}^{M \times M}\right)^m$, and signature $\sigma \in (\{0, 1\}^\lambda \times \mathbb{F}_q^n) \cup \{\perp\}$

Output: accept or reject

```

1: if  $\sigma = \perp$  then
2:   return reject
3: end if
4:  $(\text{seed}_{\text{pk}}, \{\bar{P}_{i,3}\}_{i \in [m]}) \leftarrow \text{pk}$ 
5:  $(\rho, \mathbf{s}) \leftarrow \sigma$ 
6: if  $-\mathbf{s}$  is lexicographically smaller than  $\mathbf{s}$  then
7:   return reject
8: end if
9: for  $i$  from 1 to  $m$  do
10:   $(\bar{P}_{i,1}, \bar{P}_{i,2}) \leftarrow \text{Expand}_{\text{pk}}(\text{seed}_{\text{pk}}, i)$ 
11:   $P_i \leftarrow W^{(N)} \phi \begin{pmatrix} \bar{P}_{i,1} & \bar{P}_{i,2} \\ \bar{P}_{i,1}^\top & \bar{P}_{i,3} \end{pmatrix} \quad \triangleright P_i \in \mathbb{F}_q^{n \times n}$ 
12: end for
13:  $\mu \leftarrow \text{SHAKE256}(\text{seed}_{\text{pk}} \parallel \text{BytesToBits}(\mathbf{M}), 512)$ 
14:  $\mathbf{t} \leftarrow \text{Hash}(\mu, \rho)$ 
15:  $\mathbf{t}' \leftarrow (\mathbf{s}^\top P_1 \mathbf{s}, \dots, \mathbf{s}^\top P_m \mathbf{s})^\top \quad \triangleright \mathbf{t}' \in \mathbb{F}_q^m$ 
16: return accept if  $\mathbf{t} = \mathbf{t}'$  and reject otherwise.

```

4.4 Representation of Keys and Signature

As a preliminary step, we introduce the representation of polynomial matrices of the quotient ring. Recall that the block size of the representation matrices ℓ and an irreducible polynomial $f \in \mathbb{F}_q[x]$ with $\deg f = \ell$ are fixed. Each $g \in \mathbb{F}_q[x]/(f)$ has ℓ coefficients, allowing it to be represented as a concatenation of ℓ finite field elements, requiring $\lceil \log_2 q \rceil \cdot \ell$ bits in total. Since Φ_g^f is derived from g , it can also be represented in the same manner.

Then, we describe the representations of the public and private keys and the signature.

- The public key pk is a concatenation of byte strings representing seed_{pk} and $P_{i,3}$ ($i \in [m]$). The seed_{pk} is stored as a byte string. For $P_{i,3}$, each

block matrix $W\Phi_g^f$ in $P_{i,3}$ is represented by a polynomial g with $\lceil \log_2 q \rceil \cdot \ell$ bits. Since each $P_{i,3}$ is a symmetric matrix, only the upper triangular part needs to be stored. Therefore, each $P_{i,3}$ is represented as a bit string of length $(\lceil \log_2 q \rceil \cdot \ell) \cdot \frac{m(m+\ell)}{2\ell^2}$, and $P_{i,3}$ for all $i \in [m]$ are stored as a byte string converted from the concatenation of these bit strings.

- The private key sk is the byte string representing seed_{sk} .
- The signature σ is a concatenation of byte strings representing ρ and $\mathbf{s}$. The salt ρ is simply stored as a byte string and $\mathbf{s}$ is stored as a byte string that is converted from a bit string of length $\lceil \log_2 q \rceil \cdot n$ representing n finite field elements in $\mathbb{F}_q$.

In the verification algorithm, invalid signatures can be rejected by checking whether they contain a $\lceil \log_2 q \rceil$ -bit string representing q .

4.5 Parameter Sets

This subsection specifies the proposed QR-UOV parameter sets and summarizes the rationale and validation for each parameter. These parameter sets are proposed in accordance with security levels I, III, and V of the NIST PQC project [NIS22]. Accordingly, we set the security parameter λ as 128, 192, and 256 for the security levels I, III, and V, respectively. Table 1 lists the main parameter sets, Tables 2 and 3 provide the corresponding auxiliary parameters, and Table 4 reports the resulting public-key, private-key, and signature sizes.

Base field. For all parameter sets, we choose $q = 127$ for its implementation efficiency. The reason that we do not use a finite field of even characteristic is as follows: If q is even, in a polynomial obtained as $\mathbf{x}^\top A \mathbf{x}$ where $A \in W^{(N)} \mathcal{A}_f^{N,N}$, the coefficients corresponding to the non-diagonal components of every diagonal block are zero owing to the symmetry of $W\Phi_g^f$.

Polynomial degree and system dimension. We provide two choices of ℓ for each security level: $\ell = 3$, which is chosen as the balanced parameter set, and $\ell = 10$, which is chosen as the small-public-key parameter set. For each choice of ℓ , the parameters v and m are selected to be as small as possible while achieving the target security level. Once (v, m, ℓ) is fixed, the remaining dimensions (n, V, M, N) are derived.

Seed lengths. The seed lengths are set to $\text{sk_seed_len} = 256$ and $\text{pk_seed_len} = 128$ for each security level. The private-seed length is chosen to support the highest security level, while the shorter public-seed length is chosen to reduce the public-key overhead.

Table 1: Main parameters for security level I, III, and V

SL	q	v	m	ℓ	sk_seed_len	pk_seed_len
I	127	156	54	3	256	128
	127	540	60	10	256	128
III	127	228	78	3	256	128
	127	820	90	10	256	128
V	127	306	105	3	256	128
	127	1040	110	10	256	128

Table 2: Part 1 of auxiliary parameters

SL	(q, v, m, ℓ)	n	N	V	M	τ_{n_1}	τ_{n_2}	τ_m	Δ
I	(127, 156, 54, 3)	210	70	52	18	4267	2916	82	2
	(127, 540, 60, 10)	600	60	54	6	15135	3357	89	2
III	(127, 228, 78, 3)	306	102	76	26	9020	6123	120	2
	(127, 820, 90, 10)	910	91	82	9	34603	7600	134	2
V	(127, 306, 105, 3)	411	137	102	35	16144	11018	162	2
	(127, 1040, 110, 10)	1150	115	104	11	55473	11760	168	2

Table 3: Part 2 of auxiliary parameters

(q, ℓ)	f	W
(127, 3)	$x^3 - x - 1$	$\begin{pmatrix} J_1 & \\ & J_2 \end{pmatrix}$
(127, 10)	$x^{10} - 2x^3 - 1$	$\begin{pmatrix} J_3 & \\ & J_7 \end{pmatrix}$

Table 4: The public key, private key, and signature sizes in bytes

SL	(q, v, m, ℓ)	public key (bytes)	private key (bytes)	signature (bytes)
I	(127, 156, 54, 3)	24,256	32	200
	(127, 540, 60, 10)	11,041	32	541
III	(127, 228, 78, 3)	71,884	32	292
	(127, 820, 90, 10)	35,454	32	821
V	(127, 306, 105, 3)	173,660	32	392
	(127, 1040, 110, 10)	63,541	32	1,039

Input byte lengths for rejection sampling. The values τ_{n_1} , τ_{n_2} , and τ_m specify the numbers of pseudorandom bytes generated as input to the corresponding rejection-sampling procedures. They are chosen as the minimum lengths that bound the respective failure probabilities by $2^{-\lambda}$. Their derivation is given in Subsection 4.7.

Rank threshold. For the rank-restriction parameter Δ , we set $\Delta = 2$ for all parameter sets. Modeling L as a uniformly random matrix gives $\Pr[\text{rank}(L) < m - 2] < 2^{-62.88}$. Theorem 2, stated later in the security proof in Subsection 6.2, gives the EUF-CMA security bound and makes explicit the effect of the rank restriction on the reduction. We therefore discuss this parameter choice and its heuristic basis in detail immediately after the theorem.

Polynomial matrices and their symmetrization. The polynomial f used to define $\mathcal{A}_f$ is chosen to be irreducible. The matrix W , for which WX is symmetric for every $X \in \mathcal{A}_f$, is constructed using anti-identity matrices J_i . Here, J_i denotes the anti-identity matrix of size i (e.g. $J_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$).

Key and signature sizes. Here, we give the sizes of the public key, private key, and signature for the parameter sets. These sizes are determined as follows:

- public key: $\left(\lceil \log_2 q \rceil \cdot \frac{m^2(m+\ell)}{2\ell} + \text{pk_seed_len}\right)$ bits
- private key: sk_seed_len bits
- signature: $(\lceil \log_2 q \rceil \cdot n + \lambda)$ bits

Therefore, the sizes of public/private keys and signature are computed as shown in Table 4.

4.6 Auxiliary Functions

We introduce the following auxiliary functions. Note that these functions use `RejSamp` (Algorithm 10), which converts a byte string into a vector over $\mathbb{F}_q$, and `RejSampPRG` (Algorithm 11), which generates a pseudorandom vector over $\mathbb{F}_q$ from a seed shown in Subsection 4.7.

`Expandsk` (Algorithm 4) expands the private seed seed_{sk} to a pair $(\bar{S}', \text{seed}_{\text{pk}})$, where $\bar{S}' \in \mathbb{F}_{q^\ell}^{V \times M}$. Using `RejSampPRG`, $(v_1, \dots, v_{n_2}) \in \mathbb{F}_q^{n_2}$ with $n_2 := \ell VM$ is expanded from seed_{sk} . Then, `ExpandMatrixVxM` given in Algorithm 5 converts $(v_1, \dots, v_{n_2})$ to $\bar{S}'$. Note that `ExpandFqI` given in Algorithm 6 converts an element of $\mathbb{F}_q^\ell$ to one of $\mathbb{F}_{q^\ell}$. We sample the matrix in row-major order and sample each polynomial in $\mathbb{F}_q[x]/(f) = \mathbb{F}_{q^\ell}$ in reverse degree order from the constant term to the coefficient of $x^{\ell-1}$.

Algorithm 4 $\text{Expand}_{\text{sk}}(\text{seed}_{\text{sk}})$

Input: private seed $\text{seed}_{\text{sk}} \in \{0, 1\}^{\text{sk_seed_len}}$ **Output:** matrix $\bar{S}' \in (\mathbb{F}_{q^\ell})^{V \times M}$ and public seed $\text{seed}_{\text{pk}} \in \{0, 1\}^{\text{pk_seed_len}}$

- 1: $n_2 := \ell VM$
 - 2: $(v_1, \dots, v_{n_2}) \leftarrow \text{RejSampPRG}(\text{seed}_{\text{sk}}, 1, \tau_{n_2}, n_2)$ $\triangleright \tau_{n_2} = \tau_{q, \lambda}(n_2)$
 - 3: $\bar{S}' \leftarrow \text{ExpandMatrixVxM}(v_1, \dots, v_{n_2})$
 - 4: $\text{seed}_{\text{pk}} \leftarrow \text{PRG}_{\text{sk}}(\text{seed}_{\text{sk}}, 2, \text{pk_seed_len})$
 - 5: **return** $(\bar{S}', \text{seed}_{\text{pk}})$
-

Algorithm 5 $\text{ExpandMatrixVxM}(v_1, \dots, v_{n_2}) \mid n_2 := \ell VM$

Input: vector $(v_1, \dots, v_{n_2}) \in \mathbb{F}_q^{n_2}$ **Output:** matrix $A \in (\mathbb{F}_{q^\ell})^{V \times M}$

- 1: $k \leftarrow 1$
 - 2: **for** i from 1 to V **do**
 - 3: **for** j from 1 to M **do**
 - 4: $A_{i,j} \leftarrow \text{ExpandFql}(v_k, \dots, v_{k+\ell-1})$
 - 5: $k \leftarrow k + \ell$
 - 6: **end for**
 - 7: **end for**
 - 8: **return** A
-

Algorithm 6 $\text{ExpandFql}(v_1, \dots, v_\ell)$

Input: vector $(v_1, \dots, v_\ell) \in \mathbb{F}_q^\ell$ **Output:** polynomial $g \in \mathbb{F}_{q^\ell}$

- 1: $g \leftarrow \sum_{i=1}^{\ell} v_i \cdot x^{i-1}$
 - 2: **return** g
-

$\text{Expand}_{\text{pk}}$ (Algorithm 7) expands the public seed seed_{pk} to $\{\bar{P}_{i,1}\}_{i \in [m]}, \{\bar{P}_{i,2}\}_{i \in [m]}$ where $\bar{P}_{i,1}$ is a symmetric matrix in $\mathbb{F}_{q^\ell}^{V \times V}$ and $\bar{P}_{i,2} \in \mathbb{F}_{q^\ell}^{V \times M}$. To enable the public seed expansion to be executed in parallel, instead of generating all matrices at once, we define the function $\text{Expand}_{\text{pk}}$ as a function that produces $\bar{P}_{i,1}$ and $\bar{P}_{i,2}$ for a given counter i . Additionally, $\bar{P}_{i,1}$ and $\bar{P}_{i,2}$ are separately generated within the function to allow further parallel processing. Using RejSampPRG , $(v_1, \dots, v_{n_1}) \in \mathbb{F}_q^{n_1}$ with $n_1 := \ell V(V+1)/2$ and $(v'_1, \dots, v'_{n_2}) \in \mathbb{F}_q^{n_2}$ with $n_2 := \ell VM$ are expanded from seed_{pk} . Then, $\text{ExpandSymmetricMatrixVxV}$ given in Algorithm 8 and ExpandMatrixVxM convert these two vectors to $\bar{P}_{i,1}$ and $\bar{P}_{i,2}$, respectively. For each matrix, we sample in row-major order and sample each polynomial in $\mathbb{F}_q[x]/(f) = \mathbb{F}_{q^\ell}$ in reverse degree order. Note that for $\bar{P}_{i,1}$ we sample only the upper-triangular elements due to the symmetry.

Algorithm 7 $\text{Expand}_{\text{pk}}(\text{seed}_{\text{pk}}, i)$

Input: public seed $\text{seed}_{\text{pk}} \in \{0, 1\}^{\text{pk_seed_len}}$ and counter $i \in [m]$

Output: matrices $(\bar{P}_{i,1}, \bar{P}_{i,2}) \in (\mathbb{F}_{q^\ell})^{V \times V} \times (\mathbb{F}_{q^\ell})^{V \times M}$

- 1: $n_1 := \ell V(V+1)/2$
 - 2: $(v_1, \dots, v_{n_1}) \leftarrow \text{RejSampPRG}(\text{seed}_{\text{pk}}, 2i-1, \tau_{n_1}, n_1) \quad \triangleright \tau_{n_1} = \tau_{q,\lambda}(n_1)$
 - 3: $\bar{P}_{i,1} \leftarrow \text{ExpandSymmetricMatrixVxV}(v_1, \dots, v_{n_1})$
 - 4: $n_2 := \ell VM$
 - 5: $(v'_1, \dots, v'_{n_2}) \leftarrow \text{RejSampPRG}(\text{seed}_{\text{pk}}, 2i, \tau_{n_2}, n_2) \quad \triangleright \tau_{n_2} = \tau_{q,\lambda}(n_2)$
 - 6: $\bar{P}_{i,2} \leftarrow \text{ExpandMatrixVxM}(v'_1, \dots, v'_{n_2})$
 - 7: **return** $(\bar{P}_{i,1}, \bar{P}_{i,2})$
-

Algorithm 8 $\text{ExpandSymmetricMatrixVxV}(v_1, \dots, v_{n_1}) \mid n_1 := \ell V(V+1)/2$

Input: vector $(v_1, \dots, v_{n_1}) \in \mathbb{F}_q^{n_1}$

Output: symmetric matrix $A \in (\mathbb{F}_{q^\ell})^{V \times V}$

- 1: $k \leftarrow 1$
 - 2: **for** i from 1 to V **do**
 - 3: **for** j from 1 to V **do**
 - 4: **if** $j < i$ **then**
 - 5: $A_{i,j} \leftarrow A_{j,i}$
 - 6: **else**
 - 7: $A_{i,j} \leftarrow \text{ExpandFqI}(v_k, \dots, v_{k+\ell-1})$
 - 8: $k \leftarrow k + \ell$
 - 9: **end if**
 - 10: **end for**
 - 11: **end for**
 - 12: **return** A
-

Hash (Algorithm 9) computes the hash value $\mathbf{t} \in \mathbb{F}_q^m$ of the message representative μ and the salt ρ using SHAKE256, where the output of SHAKE256 is converted to an element of $\mathbb{F}_q^m$ by RejSamp .

Algorithm 9 $\text{Hash}(\mu, \rho)$

Input: message $\mu \in \mathbb{B}^{64}$ and salt $\rho \in \{0, 1\}^\lambda$

Output: hash value $\mathbf{t} \in \mathbb{F}_q^m$

- 1: $(r_1, \dots, r_{\tau_m}) \leftarrow \text{SHAKE256}(\text{BytesToBits}(\mu) \parallel \rho, 8\tau_m) \triangleright r_i \in \mathbb{B}, \tau_m = \tau_{q,\lambda}(m)$
 - 2: $\mathbf{t} \leftarrow \text{RejSamp}((r_1, \dots, r_{\tau_m}), \tau_m, m)$
 - 3: **return** $\mathbf{t}$
-

4.7 Generation of Pseudorandom Finite Field Elements

Elements over $\mathbb{F}_q$ are sampled within auxiliary functions shown in Subsection 4.6. A pseudorandom byte string is generated to obtain these elements. From each byte, $\lceil \log_2 q \rceil$ bits are extracted to retrieve a value in the range $[0, 2^{\lceil \log_2 q \rceil})$. In the range of $[0, 2^{\lceil \log_2 q \rceil})$, q is the only number that does not belong to $\mathbb{F}_q$, since q is chosen to be a Mersenne prime in this specification. Therefore, when q is obtained from the sequence of pseudorandom numbers, q should be skipped and not chosen. To obtain a vector of length n' , the first n' masked values are used as the initial output positions. Each occurrence of q among these positions is replaced, from left to right, by the next value different from q among the remaining masked values. If there are not enough such values to fill all these positions, rejection sampling is said to fail.

The above procedure is called *rejection sampling*. Algorithm 10 shows the algorithm **RejSamp** to obtain a vector over $\mathbb{F}_q$ from a byte string $\mathbb{B}^\tau$, while **RejSampPRG** given in Algorithm 11 generates a pseudorandom byte string using a pseudorandom generator PRG (see Subsection 4.8) and converts the byte string into a vector over $\mathbb{F}_q$ using **RejSamp**. For private key expansion, PRG is instantiated as PRG_{sk} , and for public key expansion, as PRG_{pk} . Here, the value of $\tau = \tau_{q,\lambda}(n')$ represents the number of random selections from the set $\{0, \dots, 2^{\lceil \log_2 q \rceil} - 1\}$ required to generate n' random elements over $\mathbb{F}_q$ with probability at least $1 - 2^{-\lambda}$. It is derived by

$$\tau_{q,\lambda}(n') := \min \left\{ t \in \mathbb{N} \mid P(n', t, q/2^{\lceil \log_2 q \rceil}) \leq 2^{-\lambda} \right\}.$$

$P(n', t, p)$ is the cumulative binomial distribution,

$$P(n', t, p) := \sum_{i=0}^{n'-1} \binom{t}{i} p^i (1-p)^{t-i} = I_{1-p}(t - n' + 1, n'),$$

which denotes the probability of less than n' successes in t independent Bernoulli trials of success probability p . $I_z(a, b)$ is called the regularized incomplete beta function, which is suitable for numerical evaluation.

Algorithm 10 $\text{RejSamp}((r_1, \dots, r_\tau), \tau, n')|_{\tau := \tau_{q,\lambda}(n')}$

Input: byte string $(r_1, \dots, r_\tau) \in \mathbb{B}^\tau$, byte length τ , and vector length n'

Output: vector $(v_1, \dots, v_{n'}) \in \mathbb{F}_q^{n'}$

```

1: for  $j$  from 1 to  $\tau$  do
2:    $v_j \leftarrow \text{BytesToInteger}(r_j) \bmod (q + 1)$ 
3: end for
4:  $k \leftarrow n' + 1$ 
5: while  $k < \tau + 1$  and  $v_k = q$  do
6:    $k \leftarrow k + 1$ 
7: end while
8: for  $j$  from 1 to  $n'$  do
9:   if  $v_j = q$  then
10:    if  $k < \tau + 1$  then
11:       $v_j \leftarrow v_k, k \leftarrow k + 1$ 
12:      while  $k < \tau + 1$  and  $v_k = q$  do
13:         $k \leftarrow k + 1$ 
14:      end while
15:    else
16:       $v_j \leftarrow 0$ 
17:    end if
18:  end if
19: end for
20: return  $(v_1, \dots, v_{n'})$   $\triangleright$  regard integer in  $\{0, \dots, q - 1\}$  as element of  $\mathbb{F}_q$ .

```

Algorithm 11 $\text{RejSampPRG}(\text{seed}, i, \tau, n')$

Input: seed seed , counter i , byte length τ , and vector length n'

Output: vector $(v_1, \dots, v_{n'}) \in \mathbb{F}_q^{n'}$

```

1:  $(r_1, \dots, r_\tau) \leftarrow \text{PRG}(\text{seed}, i, 8\tau)$ 
2:  $(v_1, \dots, v_{n'}) \leftarrow \text{RejSamp}((r_1, \dots, r_\tau), \tau, n')$ 
3: return  $(v_1, \dots, v_{n'})$ .

```

4.8 Pseudorandom Generator

Let PRG be a pseudorandom generator that takes as input a seed `seed`, a counter i , and an output bit length $8k$, and outputs a k -byte string. We introduce two PRG instances, denoted PRG_{sk} for private key expansion and PRG_{pk} for public key expansion. For each security level, PRG_{sk} is instantiated with SHAKE256 [FIP15] and PRG_{pk} with either SHAKE128 [FIP15] or AES128 [FIP01].

Private key expansion. $\text{PRG}_{\text{sk}}(\text{seed}, i, 8k)$ outputs $\text{out} \in \mathbb{B}^k$ computed as

$$\text{out} \leftarrow \text{SHAKE256}(\text{seed} \parallel \text{IntegerToBits}(i - 1, 16), 8k).$$

Public key expansion. PRG_{pk} is instantiated as one of the following options.

- *SHAKE option.* $\text{PRG}_{\text{pk}}(\text{seed}, i, 8k)$ outputs $\text{out} \in \mathbb{B}^k$ computed as

$$\text{out} \leftarrow \text{SHAKE128}(\text{seed} \parallel \text{IntegerToBits}(i - 1, 16), 8k).$$

- *AES option.* $\text{PRG}_{\text{pk}}(\text{seed}, i, 8k)$ outputs $\text{out} \in \mathbb{B}^k$ computed as follows:

1. $b \leftarrow \text{NULL}$
2. **for** j from 1 to $\lceil k/16 \rceil$:
 - $x \leftarrow \text{IntegerToBits}(i - 1, 64) \parallel \text{IntegerToBits}(j - 1, 64)$
 - $b \leftarrow b \parallel \text{AES128}(\text{seed}, x)$
3. $\text{out} \leftarrow \text{Trunc}_k(b)$

4.9 Randomness in Signature Generation

During the execution of `Sign` (Algorithm 2), values obtained either by random selection or by pseudorandom generation are used to choose the vinegar variables, to generate the salt, and, when the corresponding linear system has more than one solution, to select a solution. This section specifies constraints on the randomness and formats of such values.

Constraints on randomly selected values. When values used in `Sign` are randomly selected, they shall be sampled from the distributions specified for their respective purposes. Any statistical deviation from the specified distribution shall be negligible with respect to the claimed security level. For sampling elements of $\mathbb{F}_q$, rejection sampling, as described in 4.7, or an equivalent method may be used.

Constraints on pseudorandom values. When values used in `Sign` are pseudorandomly generated, they shall be generated by a pseudorandom process whose outputs are computationally indistinguishable, within the claimed security level, from independently sampled values drawn from the distributions specified for the respective purposes.

If the same seed is used to generate values for more than one purpose, each invocation shall include a distinct, unambiguously encoded domain-separation input that identifies the purpose for which the values are generated.

Format of random values for each purpose. The random or pseudorandom values used in `Sign` shall have the following forms, according to their purpose. Each value shall be uniformly sampled or generated in accordance with the requirements specified above.

- **Vinegar variables** (line 3 of `Sign`): The randomness for the vinegar variables shall be interpreted as a vector in $\mathbb{F}_q^v$, where v is the number of vinegar variables. Each component of this vector is used as one vinegar variable.
- **Salt** (line 16 of `Sign`): The salt shall be a λ -bit string. A separate salt value shall be used for each signing attempt until a valid solution is found.
- **Sampling of a solution** (line 19 of `Sign`)
Randomness for solution sampling is used only when the corresponding linear system has more than one solution. If the solution is represented using d free variables, then the random value for this purpose shall be a vector of $\mathbb{F}_q^d$. The components of this vector shall be assigned to the free variables, and the remaining variables shall be determined by the linear system.

4.10 Note on Basic Linear Algebra

Like other UOV schemes, QR-UOV also requires solving a system of linear equations to generate a signature. A tremendous amount of research exists on algorithms for solving linear equations, including well-known constant-time implementations [CKY21, BCH⁺23]. Although we did not use such an algorithm in our reference implementation, it should be employed for critical applications.

5 Performance Analysis

This section is informative. Performance results characterize the specified implementations and platforms; they do not define conformance requirements or side-channel-security claims.

Tables 5 and 6 present the benchmark results on an AVX2-capable platform. These implementations are written in C with AVX2 intrinsics. They are compiled with SUPERCOP’s default compiler options, allowing the compiler to automatically generate processor-specific instructions.

Recent work on the AVX2 implementation of QR-UOV [AUH26] has substantially improved the performance of the Round 3 implementation compared with that of the Round 2 implementation. Although a direct comparison is difficult because of differences in the experimental environments and evaluation methodologies, the reported Round 3 implementation achieves approximately a threefold speedup in signing and a 3.5-fold speedup in verification over the Round 2 implementation for the fastest parameter set.

At these performance levels, message-digest computation accounts for a significant fraction of the total signing and verification time. Therefore, throughout this document, we report the cycle count measured for zero-length messages to evaluate the performance of the cryptographic core independently of the cost of message digest computation.

SUPERCOP recently replaced the median with the stabilized substitute for the 2nd quartile (StQ2) [Ber25] as its reported statistical measure. Accordingly, throughout this document, the reported cycle counts are the averages of the StQ2 values reported by SUPERCOP.

The experimental setup is as follows.

Processor: AMD Ryzen Threadripper PRO 9995WX
Base Clock: 2.5 GHz, **Boost Clock:** up to 5.4 GHz
Memory: 64 GB (DDR5-5600 16 GB×4)
OS: Linux 6.6.87.2-microsoft-standard-WSL2 (Windows 11)
Linux Distribution: Ubuntu 26.04 LTS
Compiler: GCC 15.2.0
Library: OpenSSL 3.5.5 (27 January 2026)
Benchmark framework: supercop-20260627 (average of StQ2 [Ber25] for messages of length 0, default okcompilers, no clang)

Table 5: AVX2 implementation with the AES-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	1.767	0.520	0.353
	(127, 540, 60, 10)	5.036	2.524	1.016
III	(127, 228, 78, 3)	6.045	1.342	1.014
	(127, 820, 90, 10)	17.920	7.860	3.099
V	(127, 306, 105, 3)	16.695	3.120	2.374
	(127, 1040, 110, 10)	32.067	13.351	5.929

Table 6: AVX2 implementation with the SHAKE-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	3.968	2.718	2.538
	(127, 540, 60, 10)	10.802	8.356	6.918
III	(127, 228, 78, 3)	12.639	7.780	7.322
	(127, 820, 90, 10)	37.936	27.755	23.215
V	(127, 306, 105, 3)	32.399	18.371	17.476
	(127, 1040, 110, 10)	70.550	51.870	44.487

Tables 7-10 report results for the optimized and reference C implementations using the AES- and SHAKE-based PRGs. These implementations are written entirely in C, without using inline assembly or processor-specific intrinsics. They are compiled with SUPERCOP’s default compiler options.

Table 7: C optimized implementation with the AES-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	9.682	1.739	0.934
	(127, 540, 60, 10)	34.096	26.910	2.687
III	(127, 228, 78, 3)	30.762	4.474	2.945
	(127, 820, 90, 10)	115.487	93.333	8.831
V	(127, 306, 105, 3)	79.693	10.518	7.023
	(127, 1040, 110, 10)	224.290	183.887	16.932

Table 8: C optimized implementation with the SHAKE-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	11.862	3.937	3.121
	(127, 540, 60, 10)	40.039	32.847	8.602
III	(127, 228, 78, 3)	37.122	10.813	9.278
	(127, 820, 90, 10)	135.465	113.378	28.951
V	(127, 306, 105, 3)	94.487	25.658	22.148
	(127, 1040, 110, 10)	263.033	222.897	55.659

Table 9: C reference implementation with the AES-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	10.612	1.739	1.504
	(127, 540, 60, 10)	141.017	26.887	25.183
III	(127, 228, 78, 3)	29.142	4.466	4.321
	(127, 820, 90, 10)	728.869	93.550	86.855
V	(127, 306, 105, 3)	100.519	10.509	9.615
	(127, 1040, 110, 10)	1715.537	183.943	169.842

Table 10: C reference implementation with the SHAKE-based PRG_{pk} (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	12.797	3.935	3.700
	(127, 540, 60, 10)	147.112	32.843	31.143
III	(127, 228, 78, 3)	35.540	10.825	10.658
	(127, 820, 90, 10)	749.388	113.401	106.810
V	(127, 306, 105, 3)	115.801	25.655	24.732
	(127, 1040, 110, 10)	1753.694	222.598	208.559

6 Expected Security Strength

In this section, we begin by introducing the underlying problems and the security definitions in Subsection 6.1. Next, we provide our security proof in Subsection 6.2. Finally, we estimate the complexity of plausible attacks against our proposed parameter sets in Subsection 6.3. See Section 7 for the details of each attack.

6.1 Underlying Problems and Security Definitions

We first introduce two problems for the security proof of QR-UOV as follows:

Definition 1 (UOV problem). Let $\text{MQ}_{q,n,m}$ denote the set of random quadratic maps with n variables and m equations over $\mathbb{F}_q$ and let $\text{UOV}_{q,v,o,m}$ denote the set of public key maps of the plain UOV with v vinegar variables, o oil variables, and m equations over $\mathbb{F}_q$. The UOV problem asks to distinguish a random quadratic system from a UOV public key. If we let $\mathcal{A}$ be a UOV distinguisher algorithm, then we say the distinguishing advantage of $\mathcal{A}$ is

$$\text{Adv}_{q,v,o,m}^{\text{UOV}}(\mathcal{A}) = \left| \Pr[\mathcal{A}(\mathcal{P}) = 1 \mid \mathcal{P} \leftarrow \text{MQ}_{q,(v+o),m}] - \Pr[\mathcal{A}(\mathcal{P}) = 1 \mid \mathcal{P} \leftarrow \text{UOV}_{q,v,o,m}] \right|.$$

When a public key map is sampled from $\text{UOV}_{q,v,o,m}$, the sampling is according to the public-key distribution induced by the plain-UOV key-generation algorithm.

Definition 2 (QR-MQ problem). Let f be an irreducible polynomial with $\deg f = \ell$. We then denote by $\text{QR}_{q,n,m,\ell}$ the set of quadratic maps constructed as follows

$$\text{QR}_{q,n,m,\ell} = \left\{ (\mathbf{x}^\top P_1 \mathbf{x}, \dots, \mathbf{x}^\top P_m \mathbf{x}) : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m \mid P_1, \dots, P_m \in W^{(N)} \mathcal{A}_f^{N,N} \right\},$$

where $N = n/\ell$. For a randomly chosen $\mathcal{P} \in \text{QR}_{q,n,m,\ell}$ and $\mathbf{t} \in \mathbb{F}_q^m$, the QR-MQ problem asks to compute $\mathbf{s}$ such that $\mathcal{P}(\mathbf{s}) = \mathbf{t}$. If we let $\mathcal{A}$ be an adversary, then we say that the advantage of $\mathcal{A}$ against the QR-MQ problem is

$$\text{Adv}_{q,n,m,\ell}^{\text{QR-MQ}}(\mathcal{A}) = \Pr[\mathcal{P}(\mathbf{s}) = \mathbf{t} \mid \mathcal{P} \leftarrow \text{QR}_{q,n,m,\ell}, \mathbf{t} \leftarrow \mathbb{F}_q^m, \mathbf{s} \leftarrow \mathcal{A}(\mathcal{P}, \mathbf{t})].$$

We prove the security of QR-UOV in Subsection 6.2, assuming the advantages against the above two problems are negligible. The first assumption is originally utilized for the security of the plain UOV, and thus seems relatively well understood. In contrast, the second assumption is inherent in QR-UOV. Therefore, assessing the hardness of the QR-MQ problem is crucial for provable security.

Subsequently, we give the definition of EUF-CMA security using the retrying signing interface.

Definition 3 (EUF-CMA security). Let $\mathcal{O}$ be a random oracle, and let $\mathcal{A}$ be an adversary. For a possibly failing signing algorithm, the signing oracle $\text{Sign}^{\mathcal{O}}$ repeatedly attempts to generate a signature with fresh independent randomness until the output is not $\perp$, and returns the resulting signature. We define the advantage of $\mathcal{A}$ against the EUF-CMA game of a signature scheme $\text{DSS} = (\text{KeyGen}, \text{Sign}^{\mathcal{O}}, \text{Verify}^{\mathcal{O}})$ in the (quantum) random oracle model as

$$\text{Adv}_{\text{DSS}}^{\text{EUF-CMA}}(\mathcal{A}) = \Pr[\text{Verify}^{\mathcal{O}}(\text{pk}, m, \sigma) = 1 \mid (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(), (m, \sigma) \leftarrow \mathcal{A}^{\mathcal{O}, \text{Sign}^{\mathcal{O}}(\text{sk}, \cdot)}(\text{pk})],$$

where $\text{Sign}^{\mathcal{O}}(\text{sk}, \cdot)$ was not queried on input m . We say DSS is *EUF-CMA secure* if its advantage is negligible for any efficient adversary in the security parameter.

Our EUF-CMA security proof mainly depends on a result by Kosuge and Xagawa [KX24], which shows the EUF-CMA security of the modified UOV [SSH11] assuming the non-invertibility (INV) of the underlying trapdoor function. The INV is defined as a security property such that, given a hard-to-invert function $F : \mathcal{X} \rightarrow \mathcal{Y}$, it is computationally hard for an adversary to find a preimage $x \in \mathcal{X}$ of a random $y \in \mathcal{Y}$.

If we denote by $\text{QRUOV}_{q,v,m,\ell}$ the set of public key maps of QR-UOV with parameters (q, v, m, ℓ) , then the advantage of $\mathcal{A}$ against the INV game of the trapdoor function T underlying QR-UOV is given by

$$\text{Adv}_T^{\text{INV}}(\mathcal{A}) = \Pr[\mathcal{P}(\mathbf{s}) = \mathbf{t} \mid \mathcal{P} \leftarrow \text{QRUOV}_{q,v,m,\ell}, \mathbf{t} \leftarrow \mathbb{F}_q^m, \mathbf{s} \leftarrow \mathcal{A}(\mathcal{P}, \mathbf{t})].$$

When a public key map is sampled from $\text{QRUOV}_{q,v,m,\ell}$, the sampling is according to the public-key distribution induced by the QR-UOV key-generation algorithm.

6.2 Security Proof

The proof is divided into three parts: first, we demonstrate $\text{INV} \Rightarrow \text{EUF-CMA}$; next, we show $\text{UOV} + \text{QR-MQ} \Rightarrow \text{INV}$; and finally, we establish $\text{UOV} + \text{QR-MQ} \Rightarrow \text{EUF-CMA}$ by simply combining the two reductions¹. For readability, the proofs of Lemmas 3 and 5 are deferred to Appendix E.

6.2.1 $\text{INV} \Rightarrow \text{EUF-CMA}$

The modified-UOV reduction. As a preliminary, we recall the EUF-CMA security proof of the modified UOV signature scheme given by Kosuge and Xagawa [KX24], which serves as the basis for our security proof of QR-UOV.

Lemma 2 (Proposition 4 in the ePrint version of [KX24], $\text{INV} \Rightarrow \text{EUF-CMA}$ (Modified UOV Signature)). *For any quantum EUF-CMA adversary $\mathcal{A}_{\text{cma}}$ against the modified UOV mUOV issuing at most q_{sign} classical queries to the signing*

¹ $X \Rightarrow Y$ means that the hardness of X implies the security property Y .

oracle and q_{qro} quantum queries to the random oracle $\mathbf{H}$, there exists an INV adversary $\mathcal{B}_{\text{inv}}$ against the trapdoor function $\mathbf{T}$ such that

$$\begin{aligned} \text{Adv}_{\text{mUOV}}^{\text{EUF-CMA}}(\mathcal{A}_{\text{cma}}) &\leq (2q_{\text{qro}} + 1)^2 \text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{B}_{\text{inv}}) + \frac{3}{2} q'_{\text{sign}} \sqrt{\frac{q'_{\text{sign}} + q_{\text{qro}} + 1}{2^\lambda}} \\ &\quad + 2(q_{\text{qro}} + 2) \sqrt{\frac{q'_{\text{sign}} - q_{\text{sign}}}{2^\lambda}}, \end{aligned}$$

where q'_{sign} bounds the total number of queries to $\mathbf{H}$ made during all signing queries, and the running time of $\mathcal{B}_{\text{inv}}$ is approximately that of $\mathcal{A}_{\text{cma}}$.

To adapt Lemma 2 to QR-UOV, we use the following oracle models for the hash and key-expansion functions.

Oracle models. We model SHAKE256 by an extendable-output random oracle

$$\mathbf{H} : \{0, 1\}^* \longrightarrow \mathbb{B}^\infty.$$

For any input X and nonnegative integer k , we define

$$\text{SHAKE256}(X, 8k) := \text{Trunc}_k(\mathbf{H}(X)).$$

Thus, the same random oracle $\mathbf{H}$ models all uses of SHAKE256 in QR-UOV, including private-key expansion, computation of the message representative μ , and computation of $\text{Hash}(\mu, \rho)$. We separately model the public-key expansion function by an extendable-output random function

$$\mathbf{H}_{\text{prg}} : \{0, 1\}^{\text{pk_seed.len}} \times [2m] \longrightarrow \mathbb{B}^\infty.$$

For a public seed seed_{pk} , a counter $i \in [2m]$, and an output length $8k$, we define

$$\text{PRG}_{\text{pk}}(\text{seed}_{\text{pk}}, i, 8k) := \text{Trunc}_k(\mathbf{H}_{\text{prg}}(\text{seed}_{\text{pk}}, i)).$$

The function $\mathbf{H}_{\text{prg}}$ models the selected public-key expansion option, instantiated by either SHAKE128 or the AES128-CTR-based expander, but not by both simultaneously. For the SHAKE option, we model SHAKE128 as an extendable-output random oracle. For the AES option, we explicitly assume that the public-seed AES128-CTR-based expander can be modeled by $\mathbf{H}_{\text{prg}}$.

The QR-UOV reduction. With these definitions, we establish the following lemma.

Lemma 3 (INV $\Rightarrow$ EUF-CMA (QR-UOV)). *Let*

$$\varepsilon_{\text{rank}} := \max_{\text{sk}} \Pr_{\mathbf{y} \leftarrow \mathbb{F}_q^v} [\text{rank}(L_{\text{sk}}(\mathbf{y})) < m - \Delta].$$

For any quantum EUF-CMA adversary $\mathcal{A}_{\text{cma}}$ against QR-UOV with oracle access to $\mathbf{H}$ and $\mathbf{H}_{\text{prg}}$, issuing at most q_{sign} classical queries to the signing oracle and q_{qro} quantum queries to the random oracle $\mathbf{H}$, there exists an INV adversary $\mathcal{B}_{\text{inv}}$ against the trapdoor function $\mathbf{T}$ such that

$$\begin{aligned} \text{Adv}_{\text{QR-UOV}}^{\text{EUF-CMA}}(\mathcal{A}_{\text{cma}}) &\leq \frac{(2q_{\text{qro}} + 1)^2}{1 - q_{\text{sign}}\varepsilon_{\text{rank}}} \text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{B}_{\text{inv}}) + \frac{3}{2}q'_{\text{sign}}\sqrt{\frac{q'_{\text{sign}} + 3q_{\text{sign}} + q_{\text{qro}} + 5}{2^\lambda}} \\ &\quad + 2(q_{\text{qro}} + 2)\sqrt{\frac{q'_{\text{sign}} - q_{\text{sign}}}{2^\lambda}} + \frac{q'_{\text{sign}} + 2m + 2}{2^\lambda}, \end{aligned}$$

where q'_{sign} is an upper bound on the total number of evaluations of $\mathbf{H}$ at inputs corresponding to $\text{Hash}(\mu, \rho)$ during all signing queries.² We assume $q_{\text{sign}}\varepsilon_{\text{rank}} < 1/2$.

6.2.2 UOV + QR-MQ $\Rightarrow$ INV

We demonstrate the INV of the underlying trapdoor function of QR-UOV. Before providing the proof, we prepare a lemma showing a bijection from quadratic maps with the QR structure over $\mathbb{F}_q$ to quadratic maps over $\mathbb{F}_{q^\ell}$. We call this transformation a *pull-back method*. The correctness of this statement is trivially derived from the existence of the function ϕ defined in Subsection 4.1.

Lemma 4. *For any irreducible polynomial f with $\deg f = \ell$, there exists a bijection from $\text{QR}_{q,n,m,\ell}$ to $\text{MQ}_{q^\ell,n/\ell,m}$. Similarly, there exists a bijection from $\text{QRUOV}_{q,v,m,\ell}$ to $\text{UOV}_{q^\ell,v/\ell,m/\ell,m}$. Moreover, the second bijection preserves the public-key distributions induced by the corresponding key-generation algorithms.*

We then show the security of the INV game of QR-UOV assuming the difficulty of the UOV and QR-MQ problems.

Lemma 5 (UOV and QR-MQ $\Rightarrow$ INV). *For any quantum INV adversary $\mathcal{A}_{\text{inv}}$ against the trapdoor function $\mathbf{T}$ underlying QR-UOV, there exist adversaries $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ against the UOV problem with parameters $(q^\ell, v/\ell, m/\ell, m)$ and the QR-MQ problem with parameters $(q, (v+m), m, \ell)$, respectively, such that*

$$\text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{A}_{\text{inv}}) \leq \text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}) + \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}}),$$

where the running times of $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ are about that of $\mathcal{A}_{\text{inv}}$.

6.2.3 UOV + QR-MQ $\Rightarrow$ EUF-CMA

Combining Lemmas 3 and 5, we obtain the EUF-CMA security proof of QR-UOV.

²Here, q_{sign} counts external signing-oracle queries; the internal repetitions caused by the rank test do not evaluate $\text{Hash}(\mu, \rho)$ and hence do not affect q'_{sign} .

Theorem 2 (UOV+QR-MQ $\Rightarrow$ EUF-CMA). *Let*

$$\varepsilon_{\text{rank}} := \max_{\text{sk}} \Pr_{\mathbf{y} \leftarrow \mathbb{F}_q^{v}} [\text{rank}(L_{\text{sk}}(\mathbf{y})) < m - \Delta].$$

For any quantum EUF-CMA adversary $\mathcal{A}_{\text{cma}}$ of QR-UOV issuing at most q_{sign} classical queries to the signing oracle and q_{gro} (quantum) random oracle queries to $\mathbf{H}$, there exist adversaries $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ against the UOV problem with parameters $(q^\ell, v/\ell, m/\ell, m)$ and the QR-MQ problem with parameters $(q, (v+m), m, \ell)$, respectively, such that

$$\begin{aligned} \text{Adv}_{\text{QR-UOV}}^{\text{EUF-CMA}}(\mathcal{A}_{\text{cma}}) &\leq \frac{(2q_{\text{gro}} + 1)^2}{1 - q_{\text{sign}}\varepsilon_{\text{rank}}} \left(\text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}) + \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}}) \right) \\ &\quad + \frac{3}{2} q'_{\text{sign}} \sqrt{\frac{q'_{\text{sign}} + 3q_{\text{sign}} + q_{\text{gro}} + 5}{2^\lambda}} + 2(q_{\text{gro}} + 2) \sqrt{\frac{q'_{\text{sign}} - q_{\text{sign}}}{2^\lambda}} \\ &\quad + \frac{q'_{\text{sign}} + 2m + 2}{2^\lambda}, \end{aligned}$$

where q'_{sign} is an upper bound on the total number of evaluations of $\mathbf{H}$ at inputs corresponding to $\text{Hash}(\mu, \rho)$ during all signing queries, and we suppose $q_{\text{sign}}\varepsilon_{\text{rank}} < 1/2$. The running times of $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ are about that of $\mathcal{A}_{\text{cma}}$.

Heuristic assessment of the rank threshold. We assess the choice $\Delta = 2$ under the random-matrix heuristic. Under the additional assumption that the resulting rank-failure estimate bounds the failure probability for every secret key, we show that $q_{\text{sign}} \leq 2^{61}$ satisfies the condition required by Theorem 2. This calculation does not establish a worst-case bound over all secret keys.

We estimate the probability that the coefficient matrix L_{sk} does not satisfy the rank condition by modeling L_{sk} as a uniformly random matrix in $\mathbb{F}_q^{m \times m}$. For a uniformly random matrix $A \in \mathbb{F}_q^{m \times m}$, the probability that $\text{rank}(A) = m - a$ is

$$p_{m,q}(a) = q^{-a^2} \frac{\prod_{j=a+1}^m (1 - q^{-j})^2}{\prod_{j=1}^{m-a} (1 - q^{-j})}.$$

Therefore, the probability that A does not satisfy $\text{rank}(A) \geq m - \Delta$ is

$$\varepsilon_{\text{rank}}(q, m, \Delta) = \sum_{a=\Delta+1}^m p_{m,q}(a).$$

For the proposed parameter sets, we set $\Delta = 2$. For $q = 127$ and every proposed value of m , the resulting rank-rejection probability is approximately

$$\varepsilon_{\text{rank}}(127, m, 2) \approx 1.173 \cdot 10^{-19} < 2^{-62.88}.$$

Thus, under the random-matrix model, signature generation is rejected by the rank condition only with a very small probability.

We next clarify the relationship between the above estimate and the EUF-CMA security of QR-UOV. As shown in Theorem 2, the rank restriction introduces the reduction-loss factor $(1 - q_{\text{sign}}\varepsilon_{\text{rank}})^{-1}$ and we assume $q_{\text{sign}}\varepsilon_{\text{rank}} < \frac{1}{2}$. Note, however, that $L_{\text{sk}}(\mathbf{y})$ is the coefficient matrix of the linear system in the oil variables obtained after fixing the vinegar variables in QR-UOV. It is determined by the quotient-ring-structured secret key and the chosen vinegar variables, and is therefore not known to be uniformly distributed over $\mathbb{F}_q^{m \times m}$. Consequently, the preceding random-matrix calculation should be understood as a heuristic estimate and does not by itself establish the bound $\varepsilon_{\text{rank}} < 2^{-62.88}$.

Under the heuristic random-matrix assumption that the rank-failure probability of $L_{\text{sk}}(\mathbf{y})$ is bounded by the preceding estimate for every secret key, limiting the number of signature generations per key to $q_{\text{sign}} \leq 2^{61}$ gives

$$q_{\text{sign}}\varepsilon_{\text{rank}} < 2^{61} \cdot 2^{-62.88} = 2^{-1.88} < \frac{1}{2}.$$

Thus, under this assumption, the choice $\Delta = 2$ and the limit of 2^{61} signature generations per key satisfy the condition required by the EUF-CMA security reduction in Theorem 2.

6.3 Security Estimation of Proposed Parameters

In this subsection, we confirm that the proposed parameters in Subsection 4.5 satisfy security levels I, III, and V of the NIST PQC project by estimating the complexity of relevant attacks on QR-UOV described in Section 7.

We begin by describing the criteria used to evaluate the security levels. The security levels I, III, and V indicate that a classical attacker needs more than 2^{143} , 2^{207} , and 2^{272} classical gates to break the parameters, respectively, from the call for additional digital signature schemes [NIS22]. The number of gates required for each attack can be computed using

$$\# \text{ gates} = \# \text{ field multiplications} \cdot (2 \cdot (\log_2 q)^2 + \log_2 q).$$

Next, we provide a list of the relevant attacks on QR-UOV along with their associated complexities. As stated in Subsection 6.2, the EUF-CMA security of QR-UOV can be reduced to the difficulty of the QR-MQ problem and the UOV problems with parameters $(q^\ell, v/\ell, m/\ell, m)$, namely UOV with v/ℓ vinegar variables, m/ℓ oil variables, and m equations over $\mathbb{F}_{q^\ell}$. We consider the hybrid approach [BFP09] with Wiedemann XL (WXL) [YCBC07] and the polynomial XL (PXL) [FK24] as attackers against the (QR-)MQ problem. (See Subsection 7.2 for the relationship between the difficulty of the QR-MQ problem and the plain MQ problem.) Also, we consider the Kipnis-Shamir [KS98], reconciliation [DYC⁺08], intersection [Beu21], and rectangular MinRank [Beu21] attacks as attackers against the UOV problem with parameters $(q^\ell, v/\ell, m/\ell, m)$. In addition to these attacks, we consider the claw finding attack. The complexity of these attacks is evaluated in Table 11.

Furthermore, key recovery attacks on QR-UOV can be performed by treating QR-UOV as plain UOV with parameters (q, v, m, m) without considering its structure. In Table 12, we estimate the complexity of three key recovery attacks, the Kipnis-Shamir, reconciliation, and intersection attacks, on the plain UOV with parameters (q, v, m, m) for each proposed parameter set. (See Subsection 7.3.4 for the reason that the rectangular MinRank is not applicable to the plain UOV with parameters (q, v, m, m) .) Then, one can confirm that for each attack, the complexity of key recovery attacks on the plain UOV with (q, v, m, m) is larger than or equal to the one on the plain UOV with $(q^\ell, v/\ell, m/\ell, m)$.

Finally, we assess the security levels of the QR-UOV parameter sets. In Tables 11 and 12, for each parameter set, the upper entry shows the number of classical gates, whereas the lower entry shows the number of quantum gates. Furthermore, the values in bold indicate the complexity of the best attack against each parameter set. These tables show that the proposed parameters satisfy the requirements for each security level I, III, and V. One can confirm that our proposed parameters for security levels I and III also satisfy security levels II and IV, respectively.

Table 11: The complexity of the classical and quantum attacks, the claw finding attack, and the hybrid approach with WXL (WXL) and PXL on the MQ problem, and the Kipnis-Shamir (KS), reconciliation (Recon.), intersection (Inter.), and rectangular MinRank (RM) attacks on $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ against the proposed parameter sets

SL	(q, v, m, ℓ)	$\log_2(\#\text{gates})$ in classical (upper)/quantum (lower)						
		Claw	WXL	PXL	KS	Recon.	Inter.	RM
I	$(127, 156, 54, 3)$	201	160	150	718	164	460	158
		201	131	128	372	164	282	158
	$(127, 540, 60, 10)$	222	151	149	3308	151	3090	156
		222	106	126	1666	151	1588	156
III	$(127, 228, 78, 3)$	286	222	211	1056	231	653	219
		286	181	180	542	227	390	219
	$(127, 820, 90, 10)$	328	231	224	5058	211	4651	217
		328	169	190	2542	211	2380	217
V	$(127, 306, 105, 3)$	380	288	279	1414	291	851	277
		380	237	238	722	289	505	277
	$(127, 1040, 110, 10)$	398	281	273	6457	276	5930	280
		398	217	234	3242	276	3030	280

Table 12: The complexity of the classical and quantum key recovery attacks on $\text{UOV}(q, v, m, m)$, the Kipnis-Shamir (KS), reconciliation (Recon.), and intersection (Inter.) attacks, against the proposed parameter sets

SL	(q, v, m, ℓ)	$\log_2(\#\text{gates})$ in classical (upper)/quantum (lower)		
		KS	Recon.	Inter.
I	$(127, 156, 54, 3)$	736	421	772
		383	348	527
	$(127, 540, 60, 10)$	3378	1341	3416
		1704	1101	1862
III	$(127, 228, 78, 3)$	1073	596	1115
		553	491	753
	$(127, 820, 90, 10)$	5127	2005	5172
		2580	1643	2809
V	$(127, 306, 105, 3)$	1431	782	1477
		732	644	997
	$(127, 1040, 110, 10)$	6526	2525	6572
		3280	2068	3556

7 Analysis of Attacks against QR-UOV

This section describes relevant attacks on QR-UOV. The rest of this section is organized as follows. Subsection 7.1 explains the claw finding attack, which finds a collision point. Subsection 7.2 explains the direct attack, which directly finds a signature for a given message. Subsection 7.3 recalls known key recovery attacks on the plain UOV. Subsection 7.4 discusses the relation between the irreducibility of the polynomial f constructing the quotient ring of QR-UOV and its security. Subsection 7.5 describes a way to transform the public key matrices of QR-UOV into the extension field $\mathbb{F}_{q^\ell}$. Subsection 7.6 provides a way of transforming the public key matrices over the base field $\mathbb{F}_q$ using the quotient ring structure.

7.1 Claw Finding Attack

This subsection considers the claw finding attack which is also called the birthday attack. Recently, Saarinen [Saa24] proposed a technique to improve the complexity estimation originally given in [BCC⁺23, BCH⁺23]. We apply this technique for the complexity estimation of claw finding attack in Subsection 6.3.

For a message representative μ obtained from a message $\mathbf{M}$, an attacker computes $\mathcal{P}(\mathbf{s}_i)$ for X inputs $\{\mathbf{s}_i\}_{i \in [X]}$ and $\text{Hash}(\mu, \rho_j)$ for Y salts $\{\rho_j\}_{j \in [Y]}$. If $XY = q^m$, then there is a collision with probability $\approx 1 - e^{-1}$, and the attacker can output the signature $(\rho_j, \mathbf{s}_i)$ for the message $\mathbf{M}$. Further, as mentioned in [Saa24], we can revise this attack by utilizing the property that $\mathcal{P}(c\mathbf{s}_i) = c^2\mathcal{P}(\mathbf{s}_i)$ for any $c \in \mathbb{F}_q$. This can reduce the image of $\mathcal{P}$ and Hash where a collision must occur by a factor of $2/(q-1)$ in the case of odd q . Following [BCC⁺23, BCH⁺23], we estimate the number of gates required for this attack considering the cost of multiplication and addition in $\mathbb{F}_q$ as follows:

$$2 \left(\frac{2}{q-1} \cdot q^m \cdot m \cdot 2^{17} \cdot (2 \cdot (\log_2 q)^2 + \log_2 q) \right)^{\frac{1}{2}}. \quad (8)$$

In this estimation, we suppose that computing Hash has a bit cost of 2^{17} and applying a fast enumeration algorithm [FT23] in $\mathbb{F}_q$ to evaluate $\mathcal{P}$ successively.

For the quantum claw finding attack, it is shown that attackers with limited time will prefer the classical attacks in [JS19]. Thus, in Subsection 6.3, we estimated the complexity of the quantum claw finding attack by equation (8).

7.2 Direct Attack

This subsection describes the direct attack, which can be defined as follows: Given a public key $\mathcal{P}$ with n variables and m equations over $\mathbb{F}_q$ and a target $\mathbf{t} \in \mathbb{F}_q^m$, the direct attack tries to solve the MQ system $\mathcal{P}(\mathbf{s}) = \mathbf{t}$ to find a signature $\mathbf{s}$.

We first explain the complexity of solving the MQ system with $n \leq m$ (overdetermined), and then show a way of reducing the MQ system with $n > m$

(underdetermined) into a smaller overdetermined system. Finally, we discuss the difficulty of the QR-MQ problem to which the security of QR-UOV is reduced in Theorem 2.

Overdetermined case. We provide a way of estimating the complexity of solving the overdetermined MQ system, since an underdetermined system can be transformed into an overdetermined system by specifying $n - m$ variables without disturbing the existence of a solution with high probability. One of the best-known approaches for algebraically solving the quadratic system is the hybrid approach [BFP09], which randomly guesses k with $0 \leq k \leq n$ variables before applying an MQ solver such as F4 [Fau99], F5 [Fau02], and XL [CKPS00]. The guessing process is repeated until a solution is obtained. The complexity of this approach with the Wiedemann XL (WXL) [YCBC07], which is a variant of XL, for a classical adversary is given by

$$\min_k \left(O \left(q^k \cdot 3 \cdot \binom{n-k+2}{2} \cdot \binom{d_{reg} + n - k}{d_{reg}}^2 \right) \right), \quad (9)$$

where d_{reg} denotes the degree of regularity of the system. The degree of regularity d_{reg} for a certain class of polynomial systems called *semi-regular systems* [Bar04, BFS04, BFSY05] is known to be the degree of the first non-positive term in the following series [BFSY05, YC05]:

$$\frac{(1 - z^2)^m}{(1 - z)^{n-k+1}}. \quad (10)$$

Empirically, the public key system of UOV is considered to be a semi-regular system. Therefore, this series (10) can be used to estimate the degree of regularity. By using Grover's algorithm [Gro96], the complexity of a quantum direct attack is estimated as

$$\min_k \left(O \left(q^{k/2} \cdot 3 \cdot \binom{n-k+2}{2} \cdot \binom{d_{reg} + n - k}{d_{reg}}^2 \right) \right). \quad (11)$$

Furthermore, a new variant of the hybrid approach with XL, which is called polynomial XL (PXL), was proposed in 2021 [FK24]. This PXL reduces the complexity by performing Gaussian elimination on the matrix over a polynomial ring, and the complexity of PXL for classical and quantum attackers is given by

$$O \left(k^2 \cdot \alpha \cdot \binom{n-k+d_{reg}}{d_{reg}} \cdot \binom{n+d_{reg}}{d_{reg}} + q^k \cdot \left(\alpha^2 \cdot \binom{k+d_{reg}}{d_{reg}} + \alpha^\omega \right) \right)$$

and $O \left(k^2 \cdot \alpha \cdot \binom{n-k+d_{reg}}{d_{reg}} \cdot \binom{n+d_{reg}}{d_{reg}} + q^{\frac{k}{2}} \cdot \left(\alpha^2 \cdot \binom{k+d_{reg}}{d_{reg}} + \alpha^\omega \right) \right),$

respectively, where k is the number of guessed variables and $\omega = 2.37$ is the constant in the complexity of matrix multiplication. Furthermore, α is given by

$$\alpha = \sum_{d=0}^{d_{reg}} \max \left\{ \text{coeff} \left((1 - z)^{m-(n-k)} (1 + z)^m, z^d \right), 0 \right\},$$

where $\text{coeff}(f, t)$ denotes the coefficient of a term t in a polynomial f .

Underdetermined case. We explain a way of solving the underdetermined MQ system efficiently. Thomae and Wolf [TW12] proposed a technique for reducing the number of variables and equations when $n > m$. For $\alpha = \lfloor \frac{n}{m} \rfloor - 1$, they reduce the $(n - m + \alpha)$ variables and α equations and thereby obtain a quadratic system with $m - \alpha$ variables and equations. Furue et al. [FNT21] improved Thomae and Wolf's technique by incorporating the guessing of k variables, as in the hybrid approach, and Hashimoto [Has23] proposed an efficient technique by modifying the technique of Furue et al. to enhance efficiency. Furthermore, Asanuma et al. [ACF⁺26] generalized the variable partitioning used in Hashimoto's method. The generalized partition-based method uses nonnegative integers $l \geq 2$, $a_1, \dots, a_l$, and $b_1, \dots, b_{l-1}$ satisfying $a_1 + \dots + a_l < m$ and $0 \leq b_i < a_i$ for each $i \in [l-1]$, and we set $s := m - (a_1 + \dots + a_l)$. The applicability conditions are given as

$$\begin{cases} n - \sum_{j=1}^i a_j \geq \sum_{j=1}^i a_j \left(\sum_{w=j+1}^l a_w \right) + a_i b_i & (i \in [l-1]), \\ n - \sum_{j=1}^l a_j \geq \sum_{j=1}^{l-1} a_j \left(\sum_{w=j+1}^l a_w \right) + s. \end{cases} \quad (12)$$

Then, the complexity of the method by Asanuma et al. is given as

$$\begin{aligned} & \sum_{i=1}^{l-1} a_i \text{MQ} \left(q, b_i + \sum_{w=i+1}^l a_w, b_i + \sum_{w=i+1}^l a_w \right) + q^s \left(\text{MQ}(q, a_l, a_l) + \right. \\ & \left. \sum_{i=2}^{l-1} (a_i b_i^2 + \text{MQ}(q, a_i - b_i, a_i - b_i)) + a_1 b_1^2 + \text{MQ}(q, a_1 - b_1, s + a_1 - b_1) \right), \end{aligned} \quad (13)$$

where $\text{MQ}(q, n, m)$ denotes the complexity of solving the MQ system with n variables and m equations in $\mathbb{F}_q$. In Subsection 6.3, we confirm the security of the proposed parameters by the complexity of the hybrid approach with WXL given by equation (9) using the technique by Asanuma et al. Note that it is difficult to combine PXL and the above technique since both algorithms utilize the variables to be guessed before substituting actual values, and thus we apply the technique of Thomae and Wolf [TW12] to PXL to estimate the complexity in Subsection 6.3.

QR-MQ problem. We finally discuss the security of the QR-MQ problem. In Table 13, for a QR-MQ system, we compare the theoretical d_{reg} and experimental d_{reg} using the Magma algebra system [BCP97]. The theoretical d_{reg} is the degree of regularity as the smallest degree with a non-positive coefficient in $(1 - z^2)^m / (1 - z)^{m-k+1}$, assuming that the system is semi-regular. The

experimental d_{reg} is defined as the smallest degree d at which the codimension of the ideal generated by the system is determined solely by the solutions of the system. In our experiments, we set m to sufficiently small values to keep the computations manageable, and omit the parameter v because the excess variables are fixed in the direct attack. For the QR-MQ map with m variables, m equations, and the block size ℓ over $\mathbb{F}_q$, we fix the last k variables as in the hybrid approach. That is, the direct attack is executed on the system of m equations in $m - k$ variables. To preserve the existence of a solution, the values assigned to the k fixed variables are chosen from a valid solution of the system. Table 13 shows that the degrees of regularity obtained experimentally are the same as those obtained theoretically. These results indicate that the difficulty of solving the public key system of QR-UOV is equivalent to that of solving the randomized MQ system.

Table 13: Theoretical and experimental degree of regularity of public key system of QR-UOV obtained using the Magma algebra system [BCP97].

(q, m, ℓ, k)	theoretical d_{reg}	experimental d_{reg}
(127, 18, 3, 2)	10	10
(127, 18, 3, 3)	8	8
(127, 18, 3, 4)	7	7
(127, 15, 3, 2)	8	8
(127, 15, 3, 3)	7	7
(127, 15, 3, 4)	6	6
(127, 10, 10, 2)	6	6
(127, 10, 10, 3)	5	5
(127, 10, 10, 4)	4	4

7.3 Key Recovery Attacks on UOV

This subsection recalls some existing key recovery attacks, the Kipnis-Shamir [KS98], reconciliation [DYC⁺08], intersection [Beu21], and rectangular MinRank [Beu21] attacks. For the security of QR-UOV, these key recovery attacks can be performed on the following two problems:

- $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$,
- $\text{UOV}(q, v, m, m)$,

where $\text{UOV}(q, v, o, m)$ denotes the plain UOV with v vinegar variables, o oil variables, and m equations over $\mathbb{F}_q$. The first one corresponds to one of the underlying problems of our security proof obtained by the pull-back transformation described in Lemma 4, and the second one is enabled by ignoring the quotient ring structure of QR-UOV. This subsection describes the behavior of the key recovery attacks on $\text{UOV}(q, v, o, m)$, and thus, by substituting $(q^\ell, v/\ell, m/\ell, m)$ and (q, v, m, m) for (q, v, o, m) in the following estimations, we

can obtain the complexity of the key recovery attacks on $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ and $\text{UOV}(q, v, m, m)$, respectively.

Recall that the key recovery attacks aim to obtain the subspace $\mathcal{S}^{-1}(\mathcal{O})$ of $\mathbb{F}_q^n$, where $\mathcal{O}$ is the oil subspace defined as

$$\mathcal{O} := \{(0, \dots, 0, \alpha_1, \dots, \alpha_o)^\top \mid \alpha_i \in \mathbb{F}_q\}.$$

7.3.1 Kipnis-Shamir Attack

The Kipnis-Shamir attack [KS98] chooses two invertible matrices W_i, W_j from the set of linear combinations of the representation matrices $P_1, \dots, P_m$ for the public key. Then, it probabilistically recovers a part of the subspace $\mathcal{S}^{-1}(\mathcal{O})$ by computing the invariant subspace of $W_i^{-1}W_j$. The complexity of the Kipnis-Shamir attack is estimated as

$$O(q^{v-o-1} \cdot o^4).$$

Grover's algorithm [Gro96] can be used to reduce the complexity for a quantum adversary to

$$O\left(q^{\frac{v-o-1}{2}} \cdot o^4\right).$$

Then, the complexity of the Kipnis-Shamir attack for classical and quantum adversaries against $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ is given as

$$O(q^{v-m-\ell} \cdot (m/\ell)^4) \quad \text{and} \quad O\left(q^{\frac{v-m-\ell}{2}} \cdot (m/\ell)^4\right).$$

Furthermore, the complexity of the Kipnis-Shamir attack for classical and quantum adversaries against $\text{UOV}(q, v, m, m)$ is given as

$$O(q^{v-m-1} \cdot m^4) \quad \text{and} \quad O\left(q^{\frac{v-m-1}{2}} \cdot m^4\right).$$

7.3.2 Reconciliation Attack

The reconciliation attack [DYC⁺08] treats a vector y of $\mathcal{S}^{-1}(\mathcal{O})$ as variables and solves the quadratic system $y^\top P_i y = 0$ ($i \in [m]$). Here, the dimension of $\mathcal{S}^{-1}(\mathcal{O})$ is o , and thus if we impose affine constraints, we then solve a system of m equations in $n-o = v$ variables and still have a solution with high probability.

We show how to estimate the complexity of the reconciliation attack by considering the cases $v > m$ and $v < m$ separately. In the case of $v > m$, which is generally satisfied in UOV to prevent the Kipnis-Shamir attack, the system of equations $y^\top P_i y = 0$ has a large number of solutions. Therefore, to determine a solution uniquely, we need to solve the following system to find multiple vectors $y_1, \dots, y_k$ of $\mathcal{S}^{-1}(\mathcal{O})$:

$$\begin{cases} y_j^\top P_i y_j = 0 & (i \in [m], 1 \leq j \leq k), \\ y_j^\top P_i y_\ell = 0 & (i \in [m], 1 \leq j < \ell \leq k). \end{cases}$$

We lower bound the complexity of solving this problem by that of solving the MQ problem with v variables and v equations. In the case of $v < m$, the complexity of the reconciliation attack is estimated as that of solving a quadratic system of m equations in v variables. In both cases, we estimate the complexity of solving these problems using the hybrid approach with WXL, as given in equations (9) and (11).

Then, the complexity against $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ is given by

$$\min_k \left(O \left(q^{\ell \cdot k} \cdot 3 \cdot \binom{v/\ell - k + 2}{2} \cdot \binom{d_{reg} + v/\ell - k}{d_{reg}}^2 \right) \right),$$

where $0 \leq k \leq v/\ell$, since $v/\ell < m$. Furthermore, the complexity against $\text{UOV}(q, v, m, m)$ is given by

$$\min_k \left(O \left(q^k \cdot 3 \cdot \binom{v - k + 2}{2} \cdot \binom{d_{reg} + v - k}{d_{reg}}^2 \right) \right),$$

where $0 \leq k \leq v$, since $v > m$.

7.3.3 Intersection Attack

In [Beu21], Beullens proposed a new key recovery attack against UOV, called an intersection attack. In the case of $v < 2o$, for an integer $k \geq 2$ satisfying $k < \frac{v}{v-o}$, let $L_1, \dots, L_k$ be k invertible matrices randomly chosen from a set of linear combinations of the representation matrices $P_1, \dots, P_m$ for the public key. This attack then solves the following equations for $\mathbf{y} \in \mathbb{F}_q^n$:

$$\begin{cases} (L_j^{-1} \mathbf{y})^\top P_i (L_j^{-1} \mathbf{y}) = 0 & (i \in [m], 1 \leq j \leq k), \\ (L_j^{-1} \mathbf{y})^\top P_i (L_\ell^{-1} \mathbf{y}) = 0 & (i \in [m], 1 \leq j < \ell \leq k). \end{cases} \quad (14)$$

Note that, for a solution $\mathbf{z}$ for this system, $\mathbf{z}$ is not a vector in $\mathcal{S}^{-1}(\mathcal{O})$, but $L_j^{-1} \mathbf{z}$ is a vector in $\mathcal{S}^{-1}(\mathcal{O})$. The solution space obtained from the above equation has $ko - (k-1)v$ dimensions. Thus, its complexity is equivalent to that of solving the quadratic system with $n - (ko - (k-1)v) = kv - (k-1)o$ variables and $\binom{k+1}{2}m - 2\binom{k}{2}$ equations owing to its linear dependency. The value of k is generally chosen such that the complexity of solving the above system takes the minimum value under the condition of $k < \frac{v}{v-o}$. On the other hand, in the case of $v \geq 2o$, which is the case of the proposed parameters of QR-UOV, the intersection attack becomes a probabilistic algorithm, which solves the system of equation (14) as $k = 2$ with n variables and $(3m - 2)$ equations and one of solutions is a target vector with a probability of approximately $q^{-v+2o-1}$. Therefore, its complexity is estimated by q^{v-2o+1} times the complexity of solving the quadratic system with n variables and $(3m - 2)$ equations. We estimate the complexity of solving these problems with the complexity of the hybrid approach with WXL in equations (9) and (11).

Then, the complexity against $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ is given by

$$\min_k \left(O \left(q^{v-2m+\ell} \cdot q^{\ell \cdot k} \cdot 3 \cdot \binom{n/\ell - k + 2}{2} \cdot \binom{d_{reg} + n/\ell - k}{d_{reg}}^2 \right) \right),$$

where $0 \leq k \leq n/\ell$, since $v/\ell > 2m/\ell$. Furthermore, the complexity against $\text{UOV}(q, v, m, m)$ is given by

$$\min_k \left(O \left(q^{v-2m+1} \cdot q^k \cdot 3 \cdot \binom{n - k + 2}{2} \cdot \binom{d_{reg} + n - k}{d_{reg}}^2 \right) \right),$$

where $0 \leq k \leq n$, since $v > 2m$.

7.3.4 Rectangular MinRank Attack

The rectangular MinRank attack [Beu21] was originally proposed for the Rainbow scheme by Beullens, and it tries to solve a MinRank problem obtained by transforming the public key matrices. Furue and Ikematsu [FI23] showed that the rectangular MinRank attack is applicable to $\text{UOV}(q^\ell, V, M, m)$ which is one of the underlying problems of the security of QR-UOV. Note that we suppose that $(P_1, \dots, P_m)$, $(F_1, \dots, F_m)$, and S are matrices representing the public and private keys of UOV with parameters (q^ℓ, V, M, m) .

Before describing the rectangular MinRank attack, we introduce a way of transforming sets of matrices used in the attack. Let $(G_1, \dots, G_m)$ be a set of N -by- N matrices over $\mathbb{F}_{q^\ell}$, and $\mathbf{g}_i^{(j)}$ denotes the j -th column vector of G_i , namely,

$$G_i = \begin{pmatrix} \mathbf{g}_i^{(1)} & \mathbf{g}_i^{(2)} & \dots & \mathbf{g}_i^{(N)} \end{pmatrix} \in \mathbb{F}_{q^\ell}^{N \times N}.$$

Then, we define the new set $(\tilde{G}_1, \dots, \tilde{G}_m)$ of N -by- m matrices as follows:

$$\begin{aligned} \tilde{G}_1 &:= \begin{pmatrix} \mathbf{g}_1^{(1)} & \mathbf{g}_2^{(1)} & \dots & \mathbf{g}_m^{(1)} \end{pmatrix}, \\ &\vdots \\ \tilde{G}_m &:= \begin{pmatrix} \mathbf{g}_1^{(N)} & \mathbf{g}_2^{(N)} & \dots & \mathbf{g}_m^{(N)} \end{pmatrix}. \end{aligned}$$

We then obtain $(\tilde{P}_1, \dots, \tilde{P}_m)$ and $(\tilde{F}_1, \dots, \tilde{F}_m)$ by applying this deformation to $(P_1, \dots, P_m)$ and $(F_1, \dots, F_m)$, respectively. Further, we have

$$(\tilde{P}_1, \dots, \tilde{P}_m) = (S^\top \tilde{F}_1, \dots, S^\top \tilde{F}_m) \cdot S.$$

For the proposed parameter sets shown in Subsection 4.5, we have $m > V > M$. From this relation, it is easily seen that the deformation matrices $\tilde{F}_{V+1}, \dots, \tilde{F}_m \in \mathbb{F}_{q^\ell}^{N \times m}$ are of rank $\leq V$ since they have the following form:

$$\begin{pmatrix} *_{V \times m} \\ 0_{M \times m} \end{pmatrix}.$$

Then, there exists a linear combination of $\tilde{P}_1, \dots, \tilde{P}_N \in \mathbb{F}_{q^\ell}^{N \times m}$ whose rank is $\leq V$. The rectangular MinRank attack utilizes this property to recover the private key.

The rectangular MinRank attack tries to find a non-zero element of $\mathcal{S}^{-1}(\mathcal{O})$. As in the case of Rainbow, the rectangular MinRank attack against UOV with (q^ℓ, V, M, m) is constructed as follows. Since $\dim(\mathcal{S}^{-1}(\mathcal{O})) = M$, there exists a non-zero N -by-1 vector with the following form:

$$\mathbf{a} = (a_1, a_2, \dots, a_{V+1}, 0, \dots, 0)^\top \in \mathcal{S}^{-1}(\mathcal{O}).$$

Then, it is shown that

$$\sum_{i=1}^{V+1} a_i \tilde{P}_i = (\tilde{P}_1, \dots, \tilde{P}_N) \cdot \mathbf{a} = (S^\top \tilde{F}_1, \dots, S^\top \tilde{F}_N) \cdot (S \cdot \mathbf{a})$$

is a linear combination of $S^\top \tilde{F}_{V+1}, \dots, S^\top \tilde{F}_N$. Thus, this linear combination is of rank $\leq V$. Namely, the vector $\mathbf{a}$ gives a solution to the MinRank problem for $(\tilde{P}_1, \dots, \tilde{P}_{V+1})$ with the target rank V . Moreover, we have

$$p_1(\mathbf{a}) = \dots = p_m(\mathbf{a}) = 0.$$

As a result, the vector $\mathbf{a} = (a_1, a_2, \dots, a_{V+1}, 0, \dots, 0)^\top$ we want to find is a common solution to the following problems:

- (i) $\text{Rank} \left(\sum_{i=1}^{V+1} a_i \tilde{P}_i \right) \leq V$,
- (ii) $p_1(\mathbf{a}) = \dots = p_m(\mathbf{a}) = 0$.

Complexity analysis. We then describe the estimation of the complexity to solve the above problems (i) and (ii). This is done along Beullens' estimation [Beu21] for the rectangular MinRank attack against Rainbow. Note that the characteristic of $\mathbb{F}_{q^\ell}$ is always odd in QR-UOV. See [FI23] for more details on complexity estimation.

First, we only consider problem (i). Fix an integer m' such that $V+1 \leq m' \leq m$. Let $\tilde{P}'_i$ be the $N \times m'$ matrix obtained by removing the column vectors from $(m'+1)$ -th to m -th of $\tilde{P}_i$. We then consider applying the support minors modeling method [BBC⁺20] to the MinRank problem $(\tilde{P}'_1, \dots, \tilde{P}'_{V+1})$ with the target rank V . Let I' be the ideal in $\mathbb{F}_{q^\ell}[\mathbf{a}, \mathbf{c}]$ generated by the bilinear equations obtained from the support minors modeling, where $\mathbf{c}$ is the set of $\binom{m'}{V+1}$ minor variables. For $b \in \mathbb{N}$, set

$$R'(b) := \sum_{i=1}^b (-1)^{i+1} \binom{m'}{V+i} \binom{N+i-1}{i} \binom{V+1+b-i}{b-i}.$$

Let $I'_{b,1}$ be the subspace of $(b, 1)$ -degree homogeneous polynomials of I' in $\mathbb{F}_{q^\ell}[\mathbf{a}, \mathbf{c}]$. Then, Bardet et al. [BBC⁺20] calculated that $\dim_{\mathbb{F}_{q^\ell}} I'_{b,1} = R'(b)$ for $1 \leq b \leq V+1$.

Next, we extend our analysis to include the problem (ii). We assume that $p_1(\mathbf{a}), \dots, p_m(\mathbf{a})$ behave as a semi-regular system, where

$$\mathbf{a} = (a_1, a_2, \dots, a_{V+1}, 0, \dots, 0)^\top.$$

Let I be the ideal generated by I' and $p_1(\mathbf{a}), \dots, p_m(\mathbf{a})$, namely,

$$I := I' + \langle p_1(\mathbf{a}), \dots, p_m(\mathbf{a}) \rangle \subset \mathbb{F}_{q^\ell}[\mathbf{a}, \mathbf{c}].$$

Moreover, set

$$\begin{aligned} G'(t_1, t_2) &:= \binom{m'}{V} t_2 + \sum_{b=1}^{V+1} \left(\binom{m'}{V} \binom{V+b-1}{b} - R'(b) \right) t_1^b t_2, \\ G(t_1, t_2) &:= G'(t_1, t_2) \cdot (1 - t_1^2)^m. \end{aligned}$$

Let $b_{\min} \in \mathbb{N}$ be the minimum of b such that

$$\dim_{\mathbb{F}_{q^\ell}} I_{b,1} = \dim_{\mathbb{F}_{q^\ell}} \mathbb{F}_{q^\ell}[\mathbf{a}, \mathbf{c}]_{b,1} - 1.$$

Then, following Beullens' estimation, we can state that $b_{\min}$ is predicted by

$$b_{\min}^{(\text{predict})} := \min \{b \mid G(t_1, t_2)_{b,1} \leq 1\}, \quad (15)$$

where $G(t_1, t_2)_{b,1}$ is the coefficient of $t_1^b t_2$.

Finally, by applying to $I_{b_{\min},1}$ the bilinear XL algorithm with Wiedemann algorithm [Wie86], we can find a solution $\mathbf{a}$ to problems (i) and (ii) with the following complexity:

$$3 \binom{m'}{V}^2 \binom{V+b_{\min}-1}{b_{\min}}^2 (V+1)^2. \quad (16)$$

In [FI23], our experiments showed that $b_{\min}$ is equal to $b_{\min}^{(\text{predict})}$ for some small parameters. Thus, we use $b_{\min}^{(\text{predict})}$ instead of $b_{\min}$ to estimate the complexity of the rectangular MinRank attack against QR-UOV theoretically in Subsection 6.3.

Note that it is more efficient to apply the support minors method to rectangular matrices where the number of rows is larger than the number of columns. Thus, for the parameters with $N < m$, we estimate the complexity of the attack by applying the MinRank attack after transposing the matrices $\tilde{P}_1, \dots, \tilde{P}_N$. This operation is clearly possible since matrix rank is invariant under transposition.

Remark 1 (Truncated Polynomial Ring Attacks on UOV). This remark discusses the security of the proposed parameters against recently proposed key-recovery attacks on UOV using the truncated polynomial rings [FI26]. This attack is constructed by generalizing recently proposed attacks on UOV, the wedge product attack [Ran26] and the symmetric algebra attack [JPH⁺26].

We consider solving the system in N variables and M equations over $\mathbb{F}_q$, whose solution space contains a vector space $\mathcal{A}$ with $\dim \mathcal{A} = \alpha$. In the truncated polynomial ring framework, we choose the parameter l , which determines the truncation level of the polynomial ring, and the degree D of the Macaulay matrix with $2 \leq D \leq \alpha(p^l - 1)$. Then, the framework using the p^l -truncated polynomial ring $R^{(p^l)} = \mathbb{F}_q[x_1, \dots, x_N] / \langle x_1^{p^l}, \dots, x_N^{p^l} \rangle$ finds a solution by computing a kernel vector of the Macaulay matrix $\mathcal{M}_D^{(p^l)}$ of degree D constructed by using $R^{(p^l)}$ from the given system. For an integer $0 \leq k < \alpha$, we further apply the attack to a reduced system in $N - k$ variables whose solution space $\mathcal{A}'$ satisfies $\dim \mathcal{A}' = \alpha - k$. In [FI26], they showed that this framework can be applied to the reconciliation and intersection attacks for key recovery against UOV variants.

In Table 14, we evaluate the complexity of the reconciliation attack using the truncated polynomial ring framework against the proposed parameters. For all the proposed parameter sets, the solution space of the intersection attack has dimension 1, and since [FI26] shows that the truncated polynomial ring framework provides no improvement over standard XL in this case, we omit it from our analysis. We consider applying the key recovery attacks by regarding the public key map of QR-UOV as $\text{UOV}(q^\ell, v/\ell, m/\ell, m)$ over the extension field. The table also shows the optimal parameters l and k that minimize the complexity, as well as the degree D derived from the parameters. From our complexity evaluation, we confirm that, unlike in some other UOV-type schemes, the truncated polynomial ring framework does not improve the efficiency of the original reconciliation attack. This can be attributed to the relatively large characteristic used in QR-UOV.

Table 14: The complexity of the reconciliation attack using the p^l -truncated framework on the proposed parameters

SL	(q, v, m, ℓ)	reconciliation	
		complexity	(l, k, D)
I	(127, 156, 54, 3)	164	(1, 17, 28)
	(127, 540, 60, 10)	151	(1, 5, 22)
III	(127, 228, 78, 3)	231	(1, 25, 40)
	(127, 820, 90, 10)	211	(1, 8, 31)
V	(127, 306, 105, 3)	291	(1, 34, 49)
	(127, 1040, 110, 10)	276	(1, 10, 43)

7.4 Irreducibility of Polynomial f

The public key matrices of QR-UOV are given as block matrices, where each component is an element of $W\mathcal{A}_f$. For the security of QR-UOV, we discuss the relation between the irreducibility of the polynomial f defining $\mathcal{A}_f$ and the existence of a transformation on symmetric matrices $W\Phi_g^f$ to a specific form ma-

trix. Indeed, the security of BAC-UOV [SP20] whose public key is represented as block anti-circulant matrices was weakened by transforming anti-circulant matrices into a specific form with zero submatrices [FKI⁺20]. Therefore, we have to find f such that there exists no such transformation on $W\Phi_g^f$.

In [FIKT21], the authors provided the following three theorems for the transformation on $W\Phi_g^f$ which show the suitability of an irreducible f for QR-UOV.

Theorem 3 (Theorem 4 in [FIKT21]). *Let $f \in \mathbb{F}_q[x]$ be a **reducible** polynomial with $\deg f = \ell$ and W be an invertible matrix such that every element of $W\mathcal{A}_f$ is a symmetric matrix. If $f = f_1 \cdots f_k$ ($k \in \mathbb{N}$), where $f_1, \dots, f_k$ are distinct and irreducible, and $\deg f_1 \leq \dots \leq \deg f_k$, then there exists an invertible matrix $L \in \mathbb{F}_q^{\ell \times \ell}$ and $i \in [\ell - 1]$ such that for any $X \in W\mathcal{A}_f$,*

$$L^\top X L = \begin{pmatrix} *_{i \times i} & 0_{i \times (\ell-i)} \\ 0_{(\ell-i) \times i} & *_{(\ell-i) \times (\ell-i)} \end{pmatrix}.$$

Theorem 4 (Theorem 5 in [FIKT21]). *With the same notation as in Theorem 3, if there exists $f' \in \mathbb{F}_q[x]$ such that $f'^2 \mid f$, there exists an invertible matrix $L \in \mathbb{F}_q^{\ell \times \ell}$ such that, for any $X \in W\mathcal{A}_f$,*

$$(L^\top X L)_{\ell\ell} = 0.$$

Theorem 5 (Theorem 2 in [FIKT21]). *Let $f \in \mathbb{F}_q[x]$ be an **irreducible** polynomial with $\deg f = \ell$ and W be an invertible matrix such that every element of $W\mathcal{A}_f$ is a symmetric matrix. Then, there is no invertible matrix $L \in \mathbb{F}_q^{\ell \times \ell}$ and $i, j \in [\ell]$ such that for any $X \in W\mathcal{A}_f$,*

$$(L^\top X L)_{i,j} = 0.$$

Theorems 3 and 4 show that if f is reducible, for any $X \in W\mathcal{A}_f$, X can be transformed into a matrix with zero submatrices by multiplying an invertible matrix and its transposition from both sides. In contrast, Theorem 5 shows that if f is irreducible, there exists no such transformation on $W\Phi_g^f$. Therefore, we select an irreducible polynomial as the f of $\mathcal{A}_f$ used in our proposed QR-UOV.

7.5 Lifting Method over Extension Field

The lifting method was proposed in [FIKT21] as a method of attacking QR-UOV by diagonalizing the matrices in $\mathcal{A}_f$ over the extension field $\mathbb{F}_{q^\ell}$. In this subsection, we show that the lifting method is essentially the same as the pull-back method given in Subsection 4.1. As a preliminary step, we introduce the following theorem.

Theorem 6 (Theorem 3 in [FIKT21]). *With the same notation as in Theorem 5,*

(i) There exists an invertible matrix $L \in \mathbb{F}_{q^\ell}^{\ell \times \ell}$ such that

$$L^{-1} \Phi_x^f L = \begin{pmatrix} x & & & \\ & x^q & & \\ & & x^{q^2} & \\ & & & \ddots \\ & & & & x^{q^{\ell-1}} \end{pmatrix}.$$

In particular, this L diagonalizes any matrix in $\mathcal{A}_f$.

(ii) The matrix L described in (i) satisfies the condition that $L^\top W L$ is diagonal. Therefore, we can write

$$L^\top W L = \begin{pmatrix} \alpha_0 & & & \\ & \alpha_1 & & \\ & & \ddots & \\ & & & \alpha_{\ell-1} \end{pmatrix}.$$

We recall the idea of the lifting method stated in [FIKT21]. The first and second statements in Theorem 6 show that for any $g \in \mathbb{F}_q[x]/(f) \cong \mathbb{F}_{q^\ell}$ the matrix $L^\top W \Phi_g^f L$ is diagonal. This indicates that $P_1, \dots, P_m$ in QR-UOV can be transformed into block diagonal matrices with block size $N \times N$. Let $L^{(N)} = I_N \otimes L$ be an $n \times n$ block diagonal matrix with block size ℓ ($n = \ell \cdot N$), for which the N diagonal blocks are L . Then, $(L^{(N)})^\top P_i L^{(N)}$ ($i \in [m]$) become block matrices wherein every component is in a diagonal form. Furthermore, there exists a permutation matrix A such that $(L^{(N)} A)^\top P_i (L^{(N)} A)$ is a block diagonal matrix with block size N , and let $\bar{L} := L^{(N)} A$. The transformed matrices $\bar{L}^\top P_i \bar{L}$ can be represented by $(\bar{L}^{-1} S \bar{L})^\top (\bar{L}^\top F_i \bar{L}) (\bar{L}^{-1} S \bar{L})$. Then, $\bar{L}^\top F_i \bar{L}$ is the diagonal concatenation of ℓ smaller matrices, similar to $\bar{L}^\top P_i \bar{L}$. Furthermore, $\bar{L}^{-1} S \bar{L}$ is also the diagonal concatenation of ℓ smaller matrices from (i) in Theorem 6. Then, owing to the structure of F_i , every diagonal block of $\bar{L}^\top F_i \bar{L}$ has an $M \times M$ zero block, similar to F_i . Therefore, each diagonal block of $\bar{L}^\top P_i \bar{L}$ has the same form as the matrix representing the public key of UOV with V vinegar variables and M oil variables over $\mathbb{F}_{q^\ell}$. The lifting method proposed in [FIKT21] executes the key recovery attacks on one of these diagonal blocks.

We show that the lifting method is essentially the same as the pull-back method. Let $A \in \mathbb{F}_q^{n \times n}$ be a permutation matrix such that

$$A^\top (X \otimes Y) A = Y \otimes X$$

for any $X \in \mathbb{F}_q^{N \times N}$ and $Y \in \mathbb{F}_q^{\ell \times \ell}$. Also, we recall the equation in Subsection 4.1

$$P_k = \sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} \otimes W \Phi_{x^i}^f.$$

Then the transformation in the above lifting method is described as follows:

$$\begin{aligned}
A^\top (L^{(N)})^\top P_k L^{(N)} A &= A^\top (L^{(N)})^\top \left(\sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} \otimes W \Phi_{x^i}^f \right) L^{(N)} A \\
&= A^\top \left(\sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} \otimes L^\top W \Phi_{x^i}^f L \right) A = A^\top \left(\sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} \otimes L^\top W L \cdot L^{-1} \Phi_{x^i}^f L \right) A \\
&= A^\top \left(\sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} \otimes \begin{pmatrix} \alpha_0 x^i & & & \\ & \alpha_1 x^{qi} & & \\ & & \ddots & \\ & & & \alpha_{\ell-1} x^{q^{\ell-1}i} \end{pmatrix} \right) A \\
&= \sum_{i=0}^{\ell-1} \begin{pmatrix} \alpha_0 x^i & & & \\ & \alpha_1 x^{qi} & & \\ & & \ddots & \\ & & & \alpha_{\ell-1} x^{q^{\ell-1}i} \end{pmatrix} \otimes \bar{P}_k^{(i)} \\
&= \begin{pmatrix} \alpha_0 \sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} x^i & & & \\ & \alpha_1 \sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} x^{qi} & & \\ & & \ddots & \\ & & & \alpha_{\ell-1} \sum_{i=0}^{\ell-1} \bar{P}_k^{(i)} x^{q^{\ell-1}i} \end{pmatrix} \\
&= \begin{pmatrix} \alpha_0 \bar{P}_k & & & \\ & \alpha_1 \bar{P}_{k,q} & & \\ & & \ddots & \\ & & & \alpha_{\ell-1} \bar{P}_{k,q^{\ell-1}} \end{pmatrix}.
\end{aligned}$$

Here, we have set $\bar{P}_{k,q^a} := (p_{i,j}^{q^a})_{i,j}$, where $\bar{P}_k = (p_{i,j})$. Therefore, $\bar{P}_{k,q}, \dots, \bar{P}_{k,q^{\ell-1}}$ are easily recovered from $\bar{P}_k$. Thus, when we consider a key recovery attack using the lifting method, it is enough to treat only $\bar{P}_k$ ($k \in [m]$). Since the pull-back method is also used to execute a key recovery attack on $\bar{P}_k$, we conclude that a key recovery attack using the pull-back method is the same as that using the lifting method. The only difference from the pull-back method is that we can apply the direct attack on the system of $\bar{L}^\top P_i \bar{L}$ ($i \in [m]$) obtained by applying the lifting method. However, for most cases, this lifting direct attack is not more efficient than the plain direct attack, since the large finite field $\mathbb{F}_{q^\ell}$ disturbs guessing some variables in the hybrid approach. Therefore, we list only the complexity of the plain direct attack in Subsection 6.3.

Remark 2. The attacks on SNOVA recently proposed by [CLVVP25] combine a transformation into the extension field and some existing attacks. One can confirm that the transformation used in the attacks on SNOVA is equivalent to the lifting method. As mentioned above, we confirmed that the attack on the lifted QR-UOV does not weaken the security of QR-UOV. Therefore, these attacks on SNOVA do not affect the security of QR-UOV.

7.6 Multiplication of Public Key Matrices by $(\Phi_x^f)^{(N)}$

This subsection introduces a technique to increase the number of public key matrices by multiplying $(\Phi_x^f)^{(N)}$ and demonstrates that this technique cannot be mounted as an effective attack on QR-UOV. The technique is based on the following observation, which is primarily derived from the method used in the attack on SNOVA [IA24]. When we multiply the public-key matrices P_i with $i \in [m]$ on the right by $(\Phi_x^f)^{(N)} \in \mathbb{F}_q^{n \times n}$, we have

$$\begin{aligned} P_i \cdot (\Phi_x^f)^{(N)} &= S^\top \cdot F_i \cdot S \cdot (\Phi_x^f)^{(N)} \\ &= S^\top \cdot F_i \cdot (\Phi_x^f)^{(N)} \cdot S, \end{aligned}$$

due to the commutativity of Φ_g^f with $g \in \mathbb{F}_q[x]/(f)$. Since $(\Phi_x^f)^{(N)}$ is a block diagonal matrix, the lower-right $m \times m$ submatrix of $F_i \cdot (\Phi_x^f)^{(N)}$ is the zero matrix as in the original F_i .

From the above observation, we can use $m \cdot \ell$ matrices $P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)}$ with $i \in [m]$ and $j \in [\ell]$ in applying the key recovery attacks. Note that it is enough to consider $m \cdot \ell$ matrices $P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)}$ since we have

$$(\Phi_{x^a}^f)^{(N)\top} \cdot P_i \cdot (\Phi_{x^b}^f)^{(N)} = P_i \cdot (\Phi_{x^{a+b}}^f)^{(N)}.$$

By applying this method, we can use more information to recover the private key compared to the original key recovery attack on UOV(q, v, m, m) over the base field $\mathbb{F}_q$.

Subsequently, we show that the method of increasing the number of public key matrices is not more efficient than the key recovery attacks over the extension field $\mathbb{F}_{q^\ell}$. More specifically, we show that the matrices $P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)}$ are essentially the same as the matrices obtained by naturally transforming the matrices $\bar{P}_i \in \mathbb{F}_{q^\ell}^{N \times N}$ in Subsection 4.1 onto the base field $\mathbb{F}_q$. For a variable set $\bar{\mathbf{y}} = (\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_N)^\top$, we consider the following m quadratic polynomials over $\mathbb{F}_{q^\ell}$:

$$\bar{\mathbf{y}}^\top \cdot \bar{P}_i \cdot \bar{\mathbf{y}} \quad (i \in [m]). \quad (17)$$

These form a public key of an instance of UOV(q^ℓ, V, M, m). For each $\bar{\mathbf{y}}_j$, we set $\bar{\mathbf{y}}_j = \sum_{k=0}^{\ell-1} \mathbf{y}_k^{(j)} x^k$, that is,

$$\bar{\mathbf{y}}_j = \begin{pmatrix} 1 & x & \dots & x^{\ell-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{y}_0^{(j)} \\ \mathbf{y}_1^{(j)} \\ \vdots \\ \mathbf{y}_{\ell-1}^{(j)} \end{pmatrix}. \quad (18)$$

Moreover, we set $\mathbf{y}^{(j)} = (\mathbf{y}_0^{(j)}, \dots, \mathbf{y}_{\ell-1}^{(j)})^\top$ and $\mathbf{y} = (\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(N)})^\top$. Then the polynomial $\bar{\mathbf{y}}^\top \cdot \bar{P}_i \cdot \bar{\mathbf{y}}$ is represented using the variables $\mathbf{y}$ by substituting (18) into (17). Namely, there exist ℓ quadratic polynomials $p_i^{(0)}(\mathbf{y}), \dots, p_i^{(\ell-1)}(\mathbf{y})$ in

n variables $\mathbf{y}$ over $\mathbb{F}_q$ such that

$$\bar{\mathbf{y}}^\top \cdot \bar{P}_i \cdot \bar{\mathbf{y}} = p_i^{(0)}(\mathbf{y}) + x \cdot p_i^{(1)}(\mathbf{y}) + \cdots + x^{\ell-1} \cdot p_i^{(\ell-1)}(\mathbf{y}). \quad (19)$$

It is clear that the polynomials $p_i^{(k)}(\mathbf{y})$ obtained in this way form a public key of $\text{UOV}(q, v, m, m \cdot \ell)$, since $\bar{P}_1, \dots, \bar{P}_m$ form a public key of $\text{UOV}(q^\ell, V, M, m)$. We show the following relation between $p_i^{(k)}(\mathbf{y})$ and $P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)}$:

Theorem 7. *For each $i \in [m]$, we have*

$$\left\langle p_i^{(0)}(\mathbf{y}), \dots, p_i^{(\ell-1)}(\mathbf{y}) \right\rangle_{\mathbb{F}_q} = \left\langle \mathbf{y}^\top \cdot P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)} \cdot \mathbf{y} \mid j \in [\ell] \right\rangle_{\mathbb{F}_q}.$$

Proof. First, we set $X = (1, x, \dots, x^{\ell-1})$. Then we obtain the equation $\bar{\mathbf{y}} = (I_N \otimes X) \cdot \mathbf{y}$. Thus, by the definition, we have the following computations:

$$\begin{aligned} \bar{\mathbf{y}}^\top \cdot \bar{P}_i \cdot \bar{\mathbf{y}} &= \mathbf{y}^\top \cdot (I_N \otimes X)^\top \cdot \bar{P}_i \cdot (I_N \otimes X) \cdot \mathbf{y} \\ &= \mathbf{y}^\top \cdot (\bar{P}_i \otimes X^\top X) \cdot \mathbf{y} \\ &= \mathbf{y}^\top \cdot \left(\sum_{h=0}^{\ell-1} x^h \bar{P}_i^{(h)} \otimes X^\top X \right) \cdot \mathbf{y} \\ &= \mathbf{y}^\top \cdot \left(\sum_{h=0}^{\ell-1} \bar{P}_i^{(h)} \otimes x^h X^\top X \right) \cdot \mathbf{y}. \end{aligned}$$

Moreover, by the result for $X^\top X$ in the lemma below, we obtain

$$\begin{aligned} &\mathbf{y}^\top \cdot \left(\sum_{h=0}^{\ell-1} \bar{P}_i^{(h)} \otimes X^\top X \cdot \Phi_{x^h}^f \right) \cdot \mathbf{y} \\ &= \mathbf{y}^\top \cdot \left(\sum_{h=0}^{\ell-1} \bar{P}_i^{(h)} \otimes \left(\sum_{k=0}^{\ell-1} x^k W \Phi_{z_k}^f \cdot \Phi_{x^h}^f \right) \right) \cdot \mathbf{y} \\ &= \sum_{k=0}^{\ell-1} x^k \cdot \mathbf{y}^\top \cdot \left(\sum_{h=0}^{\ell-1} \bar{P}_i^{(h)} \otimes W \Phi_{x^h}^f \right) \cdot (\Phi_{z_k}^f)^{(N)} \cdot \mathbf{y} \\ &= \sum_{k=0}^{\ell-1} x^k \cdot \mathbf{y}^\top \cdot P_i \cdot (\Phi_{z_k}^f)^{(N)} \cdot \mathbf{y}. \end{aligned}$$

As a result, for each i and k , we have

$$p_i^{(k)}(\mathbf{y}) = \mathbf{y}^\top \cdot P_i \cdot (\Phi_{z_k}^f)^{(N)} \cdot \mathbf{y}.$$

Since $z_0, z_1, \dots, z_{\ell-1}$ in $\mathbb{F}_{q^\ell}$ are linearly independent over $\mathbb{F}_q$ from the lemma below, the theorem holds. $\square$

Lemma 6. (i) $xX^\top X = X^\top X \cdot \Phi_x^f$.

(ii) There exist $z_0, \dots, z_{\ell-1} \in \mathbb{F}_{q^\ell}$ such that $X^\top X = \sum_{k=0}^{\ell-1} x^k W \Phi_{z_k}^f$.

(iii) $\{z_0, \dots, z_{\ell-1}\}$ are linearly independent over $\mathbb{F}_q$.

Proof. (i) It is easy to show statement (i) from the definition of Φ_x^f .

(ii) We set $X^\top X = \sum_{k=0}^{\ell-1} x^k Y_k$, where $Y_k \in \mathbb{F}_q^{\ell \times \ell}$. Since $X^\top X$ and $X^\top X \cdot \Phi_x^f$ are symmetric, Y_k and $Y_k \Phi_x^f$ are also symmetric. Thus, we have

$$W^{-1}(Y_k \Phi_x^f) = W^{-1} \Phi_x^{f^\top} Y_k = \Phi_x^f W^{-1} Y_k.$$

This implies that $W^{-1} Y_k$ and Φ_x^f are commutative. Therefore, there exists $z_k \in \mathbb{F}_{q^\ell}$ such that $W^{-1} Y_k = \Phi_{z_k}^f$.

(iii) By the definition of X ,

$$X^\top X = \begin{pmatrix} X \\ X \Phi_x^f \\ \vdots \\ X \Phi_{x^{\ell-1}}^f \end{pmatrix} = \sum_{k=0}^{\ell-1} x^k \begin{pmatrix} e_{k+1} \\ e_{k+1} \cdot \Phi_x^f \\ \vdots \\ e_{k+1} \cdot \Phi_{x^{\ell-1}}^f \end{pmatrix},$$

where $e_{k+1} = (0, \dots, 0, \overset{k+1}{1}, 0, \dots, 0)$. Therefore, we have $W \Phi_{z_k}^f = \begin{pmatrix} e_{k+1} \\ e_{k+1} \cdot \Phi_x^f \\ \vdots \\ e_{k+1} \cdot \Phi_{x^{\ell-1}}^f \end{pmatrix}$.

Since the first row of $W \Phi_{z_k}^f$ is e_{k+1} , it is clear that $W \Phi_{z_0}^f, \dots, W \Phi_{z_{\ell-1}}^f$ are linearly independent. Thus, (iii) holds. $\square$

From Theorem 7, we see that the $\mathbb{F}_q$ -vector space spanned by $m \cdot \ell$ polynomials $\mathbf{y}^\top \cdot P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)} \cdot \mathbf{y}$ discussed above is the same as that spanned by the public key $p_i^{(k)}(\mathbf{y})$ of UOV $(q, v, m, m \cdot \ell)$ obtained from a public key of UOV (q^ℓ, V, M, m) by transforming onto the base field $\mathbb{F}_q$. At present, no effective attacks on UOV over $\mathbb{F}_{q^\ell}$ are known that exploit the transformation of its public key into polynomials over the base field $\mathbb{F}_q$. Indeed, we confirmed that the key recovery attacks on the polynomials $p_i^{(k)}(\mathbf{y})$ for our proposed parameter sets are not efficient. In conclusion, the key recovery attacks using $P_i \cdot (\Phi_{x^{j-1}}^f)^{(N)}$ or $p_i^{(k)}(\mathbf{y})$ do not weaken the security of QR-UOV.

8 Advantages and Limitations

The main advantages of QR-UOV are as follows:

- **Public key and signature sizes.** The plain UOV is known as a scheme with a small signature and a large public key. (The second round parameters in security level I have approximately 40KB public key and 100B signature [BCD⁺25].) Our proposed parameter sets with $q = 127$ and $\ell = 3$ reduce the public key size by approximately 50% compared with the plain UOV (approximately 20KB in security level I) with signatures of a few hundred bytes.
- **Efficiency.** The signature generation and verification processes consist of simple linear algebra operations over small finite fields (e.g. $\mathbb{F}_{127}$), and thus QR-UOV can be implemented very efficiently.
- **Security.** The EUF-CMA security of QR-UOV is formally proven in the QROM assuming the difficulty of two problems, the UOV and QR-MQ problems. The security of the plain UOV is based on the UOV problem, and thus it seems relatively well understood. In contrast, the QR-MQ problem is a new assumption introduced for QR-UOV. Though there exists no formal reduction from the QR-MQ problem into the plain MQ problem, we provide some experimental facts that indicate that the difficulty of solving the QR-MQ problem is equivalent to that of solving the plain MQ problem.
- **Simplicity.** The design of the plain UOV is extremely simple, and QR-UOV is a natural extension of UOV using a quotient-ring structure. The transition from UOV to QR-UOV can be viewed as analogous to the transition from LWE to MLWE, where the MLWE problem generalizes LWE by using a module of vectors over a ring. Therefore, it requires only a minimum knowledge of algebra to understand and implement QR-UOV.

The main disadvantage of QR-UOV is the large size of the public key compared with other post-quantum signature schemes such as lattice-based signatures. As we mentioned above, the public key size of QR-UOV is reduced from that of the plain UOV. However, some selected lattice-based signature schemes have even smaller public keys with approximately 1000B in security level I. This disadvantage might make it difficult to apply QR-UOV to constrained devices such as smart cards. However, the increase in memory capabilities in the future will relax the impact of this disadvantage.

References

- [ACF⁺26] Hideki Asanuma, Yilong Chen, Hiroki Furue, Kosuke Sakata, and Tsuyoshi Takagi. Improved complexity estimates for underdetermined MQ systems via generalized variable partitioning. *IACR Cryptology ePrint Archive*, Report 2026/1054, 2026.
- [AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In *Advances in Cryptology – CRYPTO 2019*, pages 269–295. Springer, 2019.
- [AUH26] Hiroshi Amagasa, Rei Ueno, and Naofumi Homma. AVX2 implementation of QR-UOV for modern x86 processors. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026(2):325–346, Apr. 2026. DOI: 10.46586/tches.v2026.i2.325-346.
- [Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2004.
- [BBC⁺20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Improvements of algebraic attacks for solving the rank decoding and minrank problems. In *Advances in Cryptology – ASIACRYPT 2020, Part I*, pages 507–536. Springer, 2020.
- [BCC⁺23] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. MAYO specification document. Round 1 submission to the NIST Additional Digital Signature Process, 2023.
- [BCC⁺24] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. Nibbling MAYO: Optimized implementations for AVX2 and Cortex-M4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(2):252–275, 2024.
- [BCD⁺25] Ward Beullens, Ming-Shing Chen, Jintai Ding, Boru Gong, Matthias J. Kannwischer, Jacques Patarin, Bo-Yuan Peng, Dieter Schmidt, Cheng-Jhih Shih, Chengdong Tao, and Bo-Yin Yang. UOV: Unbalanced oil and vinegar: Algorithm specifications and supporting documentation, version 2.0. Round 2 submission to the NIST Additional Digital Signature Process, 2025.
- [BCH⁺23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):321–365, 2023.

- [BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.
- [Ber25] Daniel J. Bernstein. Robust statistics for rejection-sampling timings. <https://cr.yp.to/papers/rsrst-20250720.pdf>, Jul. 2025. Online manuscript, version dated July 27, 2025. Accessed August 5, 2026.
- [Beu21] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In *Advances in Cryptology – EUROCRYPT 2021*, pages 348–373. Springer, 2021.
- [Beu22] Ward Beullens. Breaking Rainbow takes a weekend on a laptop. In *Advances in Cryptology – CRYPTO 2022*, pages 464–479. Springer, 2022.
- [BFP09] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Hybrid approach for solving multivariate systems over finite fields. *Journal of Mathematical Cryptology*, 3(3):177–197, 2009.
- [BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *International Conference on Polynomial System Solving — ICPSS 2004*, pages 71–75, 2004.
- [BFSY05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. Asymptotic behavior of the index of regularity of quadratic semi-regular polynomial systems. In *8th International Symposium on Effective Methods in Algebraic Geometry — MEGA 2005*, 2005.
- [BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.
- [CHT12] Peter Czypek, Stefan Heyse, and Enrico Thomae. Efficient implementations of MQPKS on constrained devices. In *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 374–389. Springer, 2012.
- [CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology — EUROCRYPT 2000*, pages 392–407. Springer, 2000.
- [CKY21] Tung Chou, Matthias J. Kannwischer, and Bo-Yin Yang. Rainbow on Cortex-M4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(4):650–675, 2021.

- [CLVVP25] Daniel Cabarcas, Peigen Li, Javier Verbel, and Ricardo Villanueva-Polanco. Improved attacks for SNOVA by exploiting stability under a group action. In *Advances in Cryptology – CRYPTO 2025, Part VI*, pages 358–389. Springer, 2025.
- [DKL⁺18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, 2018.
- [DS05] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *Applied Cryptography and Network Security*, pages 164–175. Springer, 2005.
- [DYC⁺08] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New differential-algebraic attacks and reparametrization of Rainbow. In *Applied Cryptography and Network Security*, pages 242–257. Springer, 2008.
- [Fau99] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
- [Fau02] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *ISSAC 2002*, pages 75–83. ACM, 2002.
- [FI23] Hiroki Furue and Yasuhiko Ikematsu. A new security analysis against MAYO and QR-UOV using rectangular MinRank attack. In *Advances in Information and Computer Security*, pages 101–116. Springer, 2023.
- [FI26] Hiroki Furue and Yasuhiko Ikematsu. Key recovery attacks on UOV using p^ℓ -truncated polynomial rings. IACR Cryptology ePrint Archive, Report 2026/298, 2026.
- [FIKT21] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. A new variant of unbalanced oil and vinegar using quotient ring: QR-UOV. In *Advances in Cryptology – ASIACRYPT 2021*, pages 187–217. Springer, 2021.
- [FIP01] Advanced encryption standard (AES). FIPS PUB 197, National Institute of Standards and Technology, 2001.
- [FIP15] SHA-3 standard: Permutation-based hash and extendable-output functions. FIPS PUB 202, National Institute of Standards and Technology, 2015.

- [FIP24] Module-lattice-based digital signature standard. FIPS PUB 204, National Institute of Standards and Technology, 2024.
- [FK24] Hiroki Furue and Momonari Kudo. Polynomial XL: A variant of the XL algorithm using Macaulay matrices over polynomial rings. In *Post-Quantum Cryptography*, pages 109–143. Springer, 2024.
- [FKI⁺20] Hiroki Furue, Koha Kinjo, Yasuhiko Ikematsu, Yacheng Wang, and Tsuyoshi Takagi. A structural attack on block-anti-circulant UOV at SAC 2019. In *Post-Quantum Cryptography*, pages 323–339. Springer, 2020.
- [FNT21] Hiroki Furue, Shuhei Nakamura, and Tsuyoshi Takagi. Improving Thomae-Wolf algorithm for solving underdetermined multivariate quadratic polynomial problem. In *Post-Quantum Cryptography*, pages 65–78. Springer, 2021.
- [FT23] Hiroki Furue and Tsuyoshi Takagi. Fast enumeration algorithm for multivariate polynomials over general finite fields. In *Post-Quantum Cryptography*, pages 357–378. Springer, 2023.
- [GHHM21] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM. In *Advances in Cryptology – ASIACRYPT 2021*, pages 637–667. Springer, 2021.
- [GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC 1996)*, pages 212–219. ACM, 1996.
- [Has23] Yasufumi Hashimoto. An improvement of algorithms to solve under-defined systems of multivariate quadratic equations. *JSIAM Letters*, 15:53–56, 2023.
- [IA24] Yasuhiko Ikematsu and Rika Akiyama. Revisiting the security analysis of SNOVA. In *Proceedings of the 11th ACM Asia Public-Key Cryptography Workshop*, pages 54–61. ACM, 2024.
- [JPH⁺26] Yi Jin, Yuansheng Pan, Xiaou He, Boru Gong, and Jintai Ding. Security analysis on UOV families with odd characteristics: Using symmetric algebra. In *Public-Key Cryptography – PKC 2026, Part I*, pages 428–454. Springer, 2026.
- [JS19] Samuel Jaques and John M. Schanck. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In *Advances in Cryptology – CRYPTO 2019*, pages 32–61. Springer, 2019.

- [KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *Advances in Cryptology — EUROCRYPT 1999*, pages 206–222. Springer, 1999.
- [KS98] Aviad Kipnis and Adi Shamir. Cryptanalysis of the oil and vinegar signature scheme. In *Advances in Cryptology — CRYPTO 1998*, pages 257–266. Springer, 1998.
- [KX24] Haruhisa Kosuge and Keita Xagawa. Probabilistic hash-and-sign with retry in the quantum random oracle model. In *Public-Key Cryptography — PKC 2024, Part I*, pages 259–288. Springer, 2024.
- [LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.
- [NIS] NIST: Post-quantum cryptography CSRC. <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization>.
- [NIS20] Status report on the second round of the NIST post-quantum cryptography standardization process. NIST Internal Report 8309, National Institute of Standards and Technology, 2020.
- [NIS22] NIST: Call for additional digital signature schemes for the post-quantum cryptography standardization process. <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf>, 2022.
- [NIS24] Status report on the first round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process. NIST Internal Report 8528, National Institute of Standards and Technology, 2024.
- [NIS26] Status report on the second round of the additional digital signature schemes for the NIST post-quantum cryptography standardization process. NIST Internal Report 8610, National Institute of Standards and Technology, 2026.
- [Ran26] Lars Ran. Wedges, oil, and vinegar. In *Advances in Cryptology — EUROCRYPT 2026, Part IV*, pages 363–388. Springer, 2026.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):1–40, 2009.
- [Saa24] Markku-Juhani O. Saarinen. Bit security of UOV 1.0 (and MAYO). Message to the NIST pqc-forum, 2024.

- [Sho99] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.
- [SP20] Alan Szepieniec and Bart Preneel. Block-anti-circulant unbalanced oil and vinegar. In *Selected Areas in Cryptography – SAC 2019*, pages 574–588. Springer, 2020.
- [SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. On provable security of UOV and HFE signature schemes against chosen-message attack. In *Post-Quantum Cryptography*, pages 68–82. Springer, 2011.
- [TW12] Enrico Thomae and Christopher Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. In *Public Key Cryptography – PKC 2012*, pages 156–171. Springer, 2012.
- [Wie86] Douglas Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.
- [YC05] Bo-Yin Yang and Jiun-Ming Chen. All in the XL family: Theory and practice. In *Information Security and Cryptology – ICISC 2004*, pages 67–86. Springer, 2005.
- [YCBC07] Bo-Yin Yang, Owen Chia-Hsin Chen, Daniel J. Bernstein, and Jiun-Ming Chen. Analysis of QUAD. In *Fast Software Encryption*, pages 290–308. Springer, 2007.

A Timing Attack Countermeasures

A.1 Overview

In this section, we describe timing-attack countermeasures for the signing algorithm in Algorithm 2. The signing algorithm contains several execution paths whose execution time may depend on intermediate values generated during signature generation. A straightforward constant-time implementation would eliminate all timing dependencies by executing every step independently of these intermediate values. However, this approach would incur considerable performance overhead.

A potential source of substantial overhead is the retry procedure for salt sampling. In particular, the probability that the linear system $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ for determining the oil variables is solvable in each trial, and consequently the number of salt sampling trials, depends strongly on the rank of its coefficient matrix L . Concealing this dependency against timing attacks would require executing a sufficiently large number of retry iterations, substantially increasing the signing time.

To avoid this overhead, we treat the rank of L as observable while protecting all remaining secret information involved in solving the linear system. More specifically, computations whose execution time may depend on the remaining secret information are revised to execute in constant time, whereas computations depending solely on the public rank are left unchanged from the original scheme.

In the following subsections, we analyze the security of revealing the rank of L and describe implementation techniques for preventing timing leakage of the remaining secret information.

A.2 Security Analysis of Rank Information

In this subsection, we discuss the security impact of leaking the rank of the coefficient matrix in the linear system solved during signature generation.

As described above, the QR-UOV signing implementation can leak the rank of the coefficient matrix $L(\mathbf{y}) \in \mathbb{F}_q^{m \times m}$ of the form

$$L(\mathbf{y}) = 2 \begin{pmatrix} \mathbf{y}^\top F_{1,2} \\ \vdots \\ \mathbf{y}^\top F_{m,2} \end{pmatrix},$$

where $\mathbf{y} \in \mathbb{F}_q^v$ denotes the vector of values assigned to the vinegar variables. Here, we investigate whether it is possible to perform an efficient attack by using the rank information of the matrix $L(\mathbf{y})$.

We consider expressing the matrix $L(\mathbf{y})$ using the public information $P_1, \dots, P_m$, a signature $\mathbf{s} = (\mathbf{s}_v \parallel \mathbf{s}_o) \in \mathbb{F}_q^n$ with $\mathbf{s}_v \in \mathbb{F}_q^v$ and $\mathbf{s}_o \in \mathbb{F}_q^m$, and a submatrix $S' \in \mathbb{F}_q^{v \times m}$ of the secret key matrix S from (5) to be recovered in the key recovery attacks. Let $L_i(\mathbf{y})$ denote the i -th row of $L(\mathbf{y})$. Then we can transform it

as follows:

$$\begin{aligned}
L_i(\mathbf{y}) &= 2(\mathbf{s}_v + S' \mathbf{s}_o)^\top (-P_{i,1} S' + P_{i,2}) \\
&= 2\mathbf{s}_v^\top (-P_{i,1} S' + P_{i,2}) + 2\mathbf{s}_o^\top (-S'^\top P_{i,1} S' + S'^\top P_{i,2}) \\
&= 2\mathbf{s}_v^\top (-P_{i,1} S' + P_{i,2}) + 2\mathbf{s}_o^\top (-P_{i,2}^\top S' + P_{i,3}) \\
&= -2(\mathbf{s}_v^\top P_{i,1} + \mathbf{s}_o^\top P_{i,2}^\top) S' + 2(\mathbf{s}_v^\top P_{i,2} + \mathbf{s}_o^\top P_{i,3}).
\end{aligned}$$

From the above expression, once the public key matrices $P_1, \dots, P_m$ and a generated signature $\mathbf{s}$ are known, each entry of $L(\mathbf{y})$ can be expressed as a linear polynomial in the unknown entries of the secret matrix S' . Consequently, if a timing attack reveals the rank of $L(\mathbf{y})$, the attacker obtains additional algebraic information about S' . We therefore consider a key-recovery attack that exploits this rank information to recover S' .

Suppose that an attacker obtains, through a timing attack, the information that the rank of the matrix $L(\mathbf{y})$ arising during the generation of a signature $\mathbf{s}$ is $m - \Delta \leq r < m$. From the transformation above, we can consider recovering the secret key by finding S' satisfying $\text{rank}(L(\mathbf{y})) = r$. This can be regarded as the MinRank problem, since each entry of $L(\mathbf{y})$ is a linear polynomial in the elements of S' . In the problem, the number of variables corresponding to S' is vm/ℓ due to the QR structure, and the target rank is r . For example, for the parameter set $(q, v, m, \ell) = (127, 156, 54, 3)$ at security level I, the resulting MinRank instance consists of $vm/\ell + 1 = 2809$ input matrices, each of size 54-by-54, with target rank 52 or 53.

We then discuss the difficulty of solving the above MinRank problem obtained from the leaked rank information. As described above, the MinRank problem has to consider equations in vm/ℓ variables over $\mathbb{F}_q$. On the other hand, in a general key-recovery attack over the extension field $\mathbb{F}_{q^\ell}$, it is sufficient to construct equations whose variables correspond only to the v/ℓ components of a single column of $\bar{S}'$. Therefore, the general key-recovery attack is substantially more efficient than solving the MinRank problem derived from the rank information of L . Moreover, since the MinRank problem treats the variables as elements over $\mathbb{F}_q$, it is difficult to combine it with a key-recovery attack over $\mathbb{F}_{q^\ell}$. These discussions indicate that, even if the rank information of $L(\mathbf{y})$ is leaked through a timing attack, it is difficult to improve the efficiency of the key recovery attack.

A.3 Implementation Remarks

In this subsection, we describe an implementation strategy for realizing the timing-attack countermeasure. Given the preceding analysis, the implementation treats the rank of L as observable and applies constant-time countermeasures to the remaining secret-dependent computations.

An analysis of the reference implementation, using TIMECOP and other analysis tools, identified the linear-system solver as the principal component

requiring protection against timing leakage. Accordingly, constant-time countermeasures are applied to the operations involved in solving the linear system in the implementation.

For concreteness, we consider a solution procedure that performs an LU decomposition, uses it to determine whether the system is consistent, and, if so, samples a solution. This procedure follows the overall structure used in the reference and optimized implementations. The modifications applied to each stage of this procedure are described below.

Data-independent pivot search. The pivot positions arising during the solution of the linear system are treated as secret information. To prevent timing leakage, the pivot search examines all candidate rows, regardless of when a valid pivot is found. The selected pivot positions are retained using mask-based assignments, without secret-dependent branches or early loop termination.

Fixed-iteration computations. Computations whose iteration ranges depend on the pivot positions in the original implementation are transformed into fixed-iteration computations. Typical examples include forward elimination and back substitution during the solution of the linear system. Instead of changing the iteration ranges according to the pivot positions, all candidate iterations are executed, and the required updates are applied using mask-based assignments.

Data-independent consistency checking. When the coefficient matrix is rank deficient, the implementation determines whether the linear system is consistent before sampling a solution. A straightforward implementation may terminate as soon as an inconsistent row is detected, but such early termination causes the execution time to depend on secret information. Instead, all rows relevant to the consistency check are processed, and the result of each check is accumulated using mask-based operations. The final decision is returned only after all rows have been examined.

Data-independent solution sampling. If the linear system has multiple solutions, random values are assigned to the free variables, and the remaining variables are set to their computed values. To prevent the processing time and memory-access pattern from depending on the pivot structure, the procedure first fills all variables with random values and then overwrites the constrained variables using mask-based assignments.

The modifications described above are based on constant-time implementation principles similar to those used in prior work on linear-system solving, such as fixed-range iterations and constant-time conditional updates [BCC⁺24]. Their concrete performance overhead in QR-UOV is evaluated below.

Performance evaluation. We evaluate the practicality of the above timing-attack countermeasure through the measurement of execution time in the same

environment used for the performance evaluation in Section 5. Table 15 reports the measured cycle counts for the implementation incorporating the countermeasure. The corresponding cycle counts for the original implementation are given in Table 9.

Compared with the original signing timings reported in Table 9, the relative overhead is larger for the parameter sets with $\ell = 3$, whereas it is substantially smaller for those with $\ell = 10$. This difference reflects the larger baseline signing cost for $\ell = 10$, which reduces the relative impact of the additional constant-time operations.

Table 15: Signing performance of the AES-PRG implementation with the proposed timing-attack countermeasures (Mcycles)

SL	(q, v, m, ℓ)	Sign with countermeasures
I	(127, 156, 54, 3)	2.916
	(127, 540, 60, 10)	28.492
III	(127, 228, 78, 3)	7.951
	(127, 820, 90, 10)	98.918
V	(127, 306, 105, 3)	18.938
	(127, 1040, 110, 10)	193.769

B Additional Parameters

This section provides the additional parameter sets of QR-UOV. We use $q = 7$ and $q = 31$ as the finite-field sizes. Table 16 provides three additional parameter sets for each security level. The sizes of public/private keys and signature are computed as shown in Table 17.

Table 16: Additional parameters for security level I, III, and V

SL	q	v	m	ℓ
I	7	740	100	10
	31	165	60	3
	31	600	70	10
III	7	1100	140	10
	31	246	87	3
	31	890	100	10
V	7	1490	190	10
	31	324	114	3
	31	1120	120	10

Table 17: The public key, private key, and signature sizes in bytes

SL	(q, v, m, ℓ)	public key (bytes)	private key (bytes)	signature (bytes)
I	(7, 740, 100, 10)	20,641	32	331
	(31, 165, 60, 3)	23,641	32	157
	(31, 600, 70, 10)	12,266	32	435
III	(7, 1100, 140, 10)	55,141	32	489
	(31, 246, 87, 3)	70,976	32	233
	(31, 890, 100, 10)	34,391	32	643
V	(7, 1490, 190, 10)	135,391	32	662
	(31, 324, 114, 3)	158,405	32	306
	(31, 1120, 120, 10)	58,516	32	807

Table 18 presents the benchmark results for the additional parameter sets. The experimental environment is identical to that described in Section 5.

Table 18: Timing results for the AVX2 implementations for additional parameters with AES PRG (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(7, 740, 100, 10)	16.015	7.675	2.523
	(31, 165, 60, 3)	2.091	0.713	0.361
	(31, 600, 70, 10)	5.942	3.232	1.053
III	(7, 1100, 140, 10)	50.807	21.703	7.077
	(31, 246, 87, 3)	8.693	1.969	0.974
	(31, 890, 100, 10)	21.242	10.001	2.981
V	(7, 1490, 190, 10)	134.179	52.744	17.656
	(31, 324, 114, 3)	20.110	3.809	2.104
	(31, 1120, 120, 10)	35.120	17.091	5.369

C Expanded QR-UOV

C.1 Overview

This appendix describes Expanded QR-UOV, an expanded-key representation intended to improve signing and verification performance when the same key is used for many operations, and presents benchmark results. In this representation, key components normally derived from seed_{pk} or seed_{sk} during signing or verification are computed in advance and stored explicitly. This increases the public- and private-key sizes but avoids repeated derivation and bit unpacking.

Expanded QR-UOV is considered primarily as a benchmarking configuration and does not define a separate required implementation. The signature format is unchanged; signing and verification follow the same mathematical procedures but use the explicitly stored components directly. Furthermore, for implementation efficiency, each $\mathbb{F}_{127}$ coefficient in the expanded keys is stored as one byte rather than using a packed 7-bit representation.

C.2 Key Representation

The standard and expanded public and private keys are represented as

$$\begin{aligned} \text{pk} &= (\text{seed}_{\text{pk}}, \{\bar{P}_{i,3}\}_{i \in [m]}), \\ \text{pk}_{\text{exp}} &= (\text{seed}_{\text{pk}}, \{\bar{P}_{i,3}\}_{i \in [m]}, \{\bar{P}_{i,1}\}_{i \in [m]}, \{\bar{P}_{i,2}\}_{i \in [m]}), \\ \text{sk} &= (\text{seed}_{\text{sk}}), \\ \text{sk}_{\text{exp}} &= (\text{seed}_{\text{sk}}, \text{seed}_{\text{pk}}, \bar{S}', \{\bar{P}_{i,1}\}_{i \in [m]}, \{\bar{F}_{i,2}\}_{i \in [m]}). \end{aligned}$$

The expanded keys explicitly store the components used during signing or verification. Each $\mathbb{F}_{q^\ell}$ element in these matrices is stored as ℓ bytes, one byte for each $\mathbb{F}_q$ coefficient with $q = 127$. For the symmetric matrices $\bar{P}_{i,1}$ and $\bar{P}_{i,3}$, only the upper-triangular entries, including the diagonal, are stored. The expanded private key stores $\bar{F}_{i,2}$ rather than $\bar{P}_{i,2}$ because line 6 of Algorithm 2 can be evaluated directly as

$$\mathbf{L}_i = 2\bar{F}_{i,2}^\top \mathbf{y} = 2W^{(M)}\phi(\bar{F}_{i,2}^\top)\mathbf{y}.$$

In the key generation algorithm, $\bar{F}_{i,2}$ is computed as

$$\bar{F}_{i,2} = -\bar{P}_{i,1}\bar{S}' + \bar{P}_{i,2}$$

and stored in sk_{exp} .

Although no seed expansion is performed when using the expanded keys, seed_{pk} is retained in both pk_{exp} and sk_{exp} because it is required to compute the message representative during verification and signing, respectively. The expanded private key also retains seed_{sk} so that it can be converted back to the standard private-key representation.

The sizes of expanded public/private keys are shown in Table 19. These sizes for the parameter sets are determined as follows:

- expanded public key: $\left(\frac{4m(v+m)(v+m+\ell)}{\ell} + \text{pk_seed_len}\right)$ bits
- expanded private key: $\left(\frac{4vm(2+v+\ell+2m)}{\ell} + \text{pk_seed_len} + \text{sk_seed_len}\right)$ bits

Table 19: Public-key, private-key, and signature sizes for Expanded QR-UOV

SL	(q, v, m, ℓ)	public key (bytes)	private key (bytes)	signature (bytes)
I	(127, 156, 54, 3)	402,586	377,724	200
	(127, 540, 60, 10)	1,098,016	1,088,688	541
III	(127, 228, 78, 3)	1,229,218	1,153,044	292
	(127, 820, 90, 10)	3,767,416	3,734,328	821
V	(127, 306, 105, 3)	2,977,711	2,790,003	392
	(127, 1040, 110, 10)	7,337,016	7,275,888	1,039

C.3 Benchmark Results

Table 20 presents the benchmark results for the implementation using the expanded keys with AVX2 intrinsics. The experimental environment is largely the same as that described in Section 5. Because the verification cycle count is very small, the measurements tend to exhibit higher variability in this setting. Therefore, we disabled CPU boost functionality (Precision Boost 2) during the measurements to improve the stability of the results. Compared with the results of the original algorithms in Table 5, Expanded QR-UOV is slightly slower in key generation because of the additional cost of producing and storing the larger keys. In contrast, signing and verification are substantially faster because the explicitly stored key components need not be derived and unpacked for each operation.

Table 20: Timing results for the AVX2 implementations of Expanded QR-UOV with AES PRG (Mcycles)

SL	(q, v, m, ℓ)	KeyGen	Sign	Verify
I	(127, 156, 54, 3)	1.949	0.203	0.074
	(127, 540, 60, 10)	5.704	0.719	0.212
III	(127, 228, 78, 3)	6.636	0.480	0.218
	(127, 820, 90, 10)	20.482	2.069	0.594
V	(127, 306, 105, 3)	18.070	1.068	0.464
	(127, 1040, 110, 10)	37.266	3.277	1.052

D QR-UOV with Full-Rank Signature Generation

In the signature generation algorithm in Subsection 4.2, we consider a variant for the case $\Delta = 0$, where a signature is generated only when L has full rank. In this variant, the linear system $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has a solution for every $\mathbf{t} \in \mathbb{F}_q^m$, and hence the repeat loop is executed only once. In this variant, when L is not full rank, the signature generation procedure is assumed to be restarted from the beginning. This variant can correspond to the second-round UOV specification, in which the salt is fixed, and new values of the vinegar variables are sampled until a full-rank matrix L is obtained.

E Proofs of Lemmas for Security Reductions

E.1 Proof of Lemma 3

Lemma 3 (restated). *Let*

$$\varepsilon_{\text{rank}} := \max_{\text{sk}} \Pr_{\mathbf{y} \xleftarrow{\$} \mathbb{F}_q^v} [\text{rank}(L_{\text{sk}}(\mathbf{y})) < m - \Delta].$$

For any quantum EUF-CMA adversary $\mathcal{A}_{\text{cma}}$ against QR-UOV with oracle access to $\mathbf{H}$ and $\mathbf{H}_{\text{prg}}$, issuing at most q_{sign} classical queries to the signing oracle and q_{qro} quantum queries to the random oracle $\mathbf{H}$, there exists an INV adversary $\mathcal{B}_{\text{inv}}$ against the trapdoor function $\mathbf{T}$ such that

$$\begin{aligned} \text{Adv}_{\text{QR-UOV}}^{\text{EUF-CMA}}(\mathcal{A}_{\text{cma}}) &\leq \frac{(2q_{\text{qro}} + 1)^2}{1 - q_{\text{sign}}\varepsilon_{\text{rank}}} \text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{B}_{\text{inv}}) + \frac{3}{2} q'_{\text{sign}} \sqrt{\frac{q'_{\text{sign}} + 3q_{\text{sign}} + q_{\text{qro}} + 5}{2^\lambda}} \\ &\quad + 2(q_{\text{qro}} + 2) \sqrt{\frac{q'_{\text{sign}} - q_{\text{sign}}}{2^\lambda}} + \frac{q'_{\text{sign}} + 2m + 2}{2^\lambda}, \end{aligned}$$

where q'_{sign} is an upper bound on the total number of evaluations of $\mathbf{H}$ at inputs corresponding to $\text{Hash}(\mu, \rho)$ during all signing queries. We assume $q_{\text{sign}}\varepsilon_{\text{rank}} < 1/2$.

Proof. To adapt the reduction of Lemma 2 to QR-UOV, we first introduce InvRejSamp as a preparation. This algorithm is needed because the signing-oracle simulator must sample a byte string uniformly from the preimage under RejSamp of a prescribed target vector.

Inverse rejection sampling. Algorithm 12 defines the inverse rejection-sampling procedure. Given a target vector over $\mathbb{F}_q$, it samples a byte string from the preimage of that vector under RejSamp . The following lemma shows that the output is uniform over this preimage.

Algorithm 12 $\text{InvRejSamp}((v_1, \dots, v_{n'}), \tau, n') \mid_{\tau := \tau_{q,\lambda}(n')}$

Input: vector $(v_1, \dots, v_{n'}) \in \mathbb{F}_q^{n'}$, byte length τ , and vector length n'

Output: byte string $(r_1, \dots, r_\tau) \in \mathbb{B}^\tau$

```

1: repeat
2:   for  $i$  from 1 to  $\tau$  do
3:      $w_i \xleftarrow{\$} \{0, \dots, q\}$ 
4:   end for
5:   Let  $(h_1, \dots, h_d)$  be the increasing sequence containing exactly the indices
      $i \in [n']$  for which  $w_i = q$ .
6:   Let  $(a_1, \dots, a_e)$  be the increasing sequence containing exactly the indices
      $i \in [\tau] \setminus [n']$  for which  $w_i < q$ .
7:   until  $e \geq d$ 
8:    $(w'_1, \dots, w'_\tau) \leftarrow (w_1, \dots, w_\tau)$ 
9:    $k \leftarrow 1$ 
10:  for  $j$  from 1 to  $n'$  do
11:    if  $w_j = q$  then
12:       $w'_{a_k} \leftarrow v_j$ 
13:       $k \leftarrow k + 1$ 
14:    else
15:       $w'_j \leftarrow v_j$ 
16:    end if
17:  end for
18:  for  $i$  from 1 to  $\tau$  do
19:     $r_i \xleftarrow{\$} \{r \in \mathbb{B} : (\text{BytesToInteger}(r) \text{ AND } q) = w'_i\}$ 
20:  end for
21:  return  $(r_1, \dots, r_\tau)$ 

```

Lemma 7. For any $(v_1, \dots, v_{n'}) \in \mathbb{F}_q^{n'}$, $\text{InvRejSamp}((v_1, \dots, v_{n'}), \tau, n')$ outputs a uniform element of

$$\{(r_1, \dots, r_\tau) \in \mathbb{B}^\tau : \text{RejSamp}((r_1, \dots, r_\tau), \tau, n') = (v_1, \dots, v_{n'}) \text{ without failure}\}.$$

Proof. Let $\mathbf{v} = (v_1, \dots, v_{n'})$ and define

$$\mathcal{R}_{\mathbf{v}} = \{\mathbf{r} \in \mathbb{B}^\tau : \text{RejSamp}(\mathbf{r}, \tau, n') = \mathbf{v} \text{ without failure}\}.$$

We first establish correctness, namely,

$$\Pr[\text{InvRejSamp}(\mathbf{v}, \tau, n') \in \mathcal{R}_{\mathbf{v}}] = 1.$$

InvRejSamp samples $(w_1, \dots, w_\tau)$ uniformly from $\{0, \dots, q\}^\tau$. Let $(h_1, \dots, h_d)$ be the increasing sequence of indices $k \in [n']$ for which $w_k = q$, and let $(a_1, \dots, a_e)$ be the increasing sequence of indices $k \in [\tau] \setminus [n']$ for which $w_k < q$. The algorithm restarts if $e < d$. Each iteration succeeds with positive probability, so

the repeat loop terminates with probability one. If $e \geq d$, the algorithm replaces each $w'_j \neq q$ among the first n' positions with v_j and replaces w'_{a_k} with v_{h_k} for every $k \in [d]$. Thus, **RejSamp** retains v_j at each non-rejected position and fills position h_k with $w'_{a_k} = v_{h_k}$. Hence, every output of **InvRejSamp** is mapped to $\mathbf{v}$ by **RejSamp** without failure.

We next establish uniformity, namely, for every $\mathbf{r} \in \mathcal{R}_{\mathbf{v}}$,

$$\Pr[\text{InvRejSamp}(\mathbf{v}, \tau, n') = \mathbf{r}] = \frac{1}{|\mathcal{R}_{\mathbf{v}}|}.$$

We call the sequence $\mathbf{w} = (w_1, \dots, w_\tau)$ sampled inside the repeat loop the *initial sequence*, and we call the sequence $\mathbf{w}' = (w'_1, \dots, w'_\tau)$ obtained after the replacements the *intermediate sequence*. We prove the above equality in three steps. First, the initial sequence is uniformly distributed over all sequences that pass the repeat loop. Second, every possible intermediate sequence is produced by the same number of initial sequences. Finally, every intermediate sequence has the same number of byte-string representations, and these representations are selected uniformly.

First, the initial sequence used in the replacement step is uniformly distributed over all initial sequences that pass the repeat loop. Indeed, every sequence in $\{0, \dots, q\}^\tau$ is initially sampled with the same probability, and the repeat loop only discards sequences for which the hole-filling procedure fails.

Second, every possible intermediate sequence is produced by exactly $q^{n'}$ initial sequences. To see this, fix an intermediate sequence from which the hole-filling procedure produces $\mathbf{v}$ without failure, and suppose that d of its first n' positions contain q . At each of the other $n' - d$ positions, the initial value may be any element of $\mathbb{F}_q$, since it is replaced by the corresponding v_j . Likewise, at each of the first d accepted positions after the first n' positions, the initial value may be any element of $\mathbb{F}_q$, since it is replaced by the corresponding v_{h_k} . All other initial values are fixed by the intermediate sequence. These choices preserve which positions contain q , so all compatible initial sequences pass the repeat loop. Therefore, their number is

$$q^{n'-d} q^d = q^{n'}.$$

This number is independent of the intermediate sequence. Together with the first statement, this shows that all possible intermediate sequences are equally likely.

Finally, every intermediate sequence has the same number of byte-string representations, and each such representation is selected with the same probability. Since q is Mersenne and $q + 1$ divides 256, each value in $\{0, \dots, q\}$ has exactly $256/(q+1)$ byte representations under the bitwise-AND operation. Hence, every intermediate sequence has exactly

$$\left(\frac{256}{q+1} \right)^\tau$$

byte-string representations. The algorithm selects each byte independently and uniformly from the corresponding set. Since all intermediate sequences are

equally likely, every byte string in $\mathcal{R}_{\mathbf{v}}$ is output with the same probability. By correctness, these probabilities sum to one over $\mathcal{R}_{\mathbf{v}}$. Therefore, for every $\mathbf{r} \in \mathcal{R}_{\mathbf{v}}$,

$$\Pr[\text{InvRejSamp}(\mathbf{v}, \tau, n') = \mathbf{r}] = \frac{1}{|\mathcal{R}_{\mathbf{v}}|}.$$

□

With this preparation in place, we now prove the reduction. First, we account for failures of **RejSamp**. We then transform the signing oracle into a secret-key-free simulation by extending the techniques used in the proof of Lemma 2. To realize this simulation for QR-UOV, we remove the rank restriction at the cost of the multiplicative factor $(1 - q_{\text{sign}}\varepsilon_{\text{rank}})^{-1}$. Finally, we show that an INV adversary can simulate the modified game by programming $\mathbf{H}_{\text{prg}}$ so that the compressed public key expands to the public map of the INV instance.

Handling rejection-sampling failure. We first modify the EUF-CMA game so that it aborts whenever fewer than n' values are accepted by **RejSamp**, instead of replacing the missing values with zero. Since the parameters are set so that the failure probability of each **RejSamp** call is bounded by $2^{-\lambda}$, we apply the union bound over all relevant calls. The function **Expand_{sk}** makes one such call, and **Expand_{pk}** makes two calls for each $i \in [m]$, giving $2m + 1$ calls for key expansion. In addition, at most q'_{sign} calls are made by **Hash** during all signing queries, and one further call is made when verifying the final forgery. Therefore, the resulting advantage gap is bounded by

$$\frac{q'_{\text{sign}} + 2m + 2}{2^{\lambda}}.$$

Signing-oracle modifications. We next modify the signing oracle to enable the reduction to INV. To explain the required simulation, we first recall the three steps from the proof of Lemma 2.

Step 1: Using tight adaptive reprogramming [GHHM21], the signing oracle repeatedly samples $(\rho, \mathbf{t})$ and reprograms the random oracle so that $\text{Hash}(\mu, \rho) = \mathbf{t}$ until $\mathbf{t}$ is successfully inverted.

Step 2: Reprogramming associated with a failed inversion is canceled. Semi-classical O2H [AHU19] ensures that the adversary cannot query $\mathbf{H}$ at the point

$$\text{BytesToBits}(\mu) \parallel \rho$$

for a salt ρ used in a failed inversion. Canceling such reprogramming therefore does not affect the adversary's view.

Step 3: To simulate signatures without the private key, the signing oracle samples a random preimage $\mathbf{s}$, reprograms the random oracle so that $\text{Hash}(\mu, \rho) = \mathcal{P}(\mathbf{s})$, and outputs $(\rho, \mathbf{s})$ as a valid signature.

For QR-UOV, we instantiate these three steps at the byte-string level as follows.

Step 1: The signing oracle repeatedly samples $\rho \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathbf{r} \xleftarrow{\$} \mathbb{B}^{\tau_m}$, sets $\mathbf{t} = \text{RejSamp}(\mathbf{r}, \tau_m, m)$, and reprograms H so that

$$\text{SHAKE256}(\text{BytesToBits}(\mu) \parallel \rho, 8\tau_m) = \mathbf{r},$$

until $\mathbf{t}$ is successfully inverted.

Step 2: Reprogramming associated with a failed inversion is canceled, as in the original proof. After this modification, the signing oracle operates as follows.

1. Repeatedly sample $\mathbf{r}$ until $\mathbf{t} = \text{RejSamp}(\mathbf{r}, \tau_m, m)$ can be inverted, and let $\mathbf{s}$ be the resulting preimage, so that $\mathbf{t} = \mathcal{P}(\mathbf{s})$.
2. Reprogram H so that $\text{SHAKE256}(\text{BytesToBits}(\mu) \parallel \rho, 8\tau_m) = \mathbf{r}$, where ρ is chosen uniformly at random.
3. If $-\mathbf{s}$ is lexicographically smaller than $\mathbf{s}$, replace $\mathbf{s}$ by $-\mathbf{s}$.
4. Output $(\rho, \mathbf{s})$.

Step 3: The secret-key-free signing-oracle simulation proceeds as follows:

1. Choose $\mathbf{s}$ randomly from $\mathbb{F}_q^n$ and let $\mathbf{t} = \mathcal{P}(\mathbf{s})$.
2. Set $\mathbf{r} \leftarrow \text{InvRejSamp}(\mathbf{t}, \tau_m, m)$; by Lemma 7, $\mathbf{r}$ is uniform over

$$\mathcal{R}_{\mathbf{t}} = \{\mathbf{r} \in \mathbb{B}^{\tau_m} : \text{RejSamp}(\mathbf{r}, \tau_m, m) = \mathbf{t} \text{ without failure}\}.$$

3. Reprogram H so that

$$\text{SHAKE256}(\text{BytesToBits}(\mu) \parallel \rho, 8\tau_m) = \mathbf{r},$$

where ρ is chosen uniformly at random.

4. If $-\mathbf{s}$ is lexicographically smaller than $\mathbf{s}$, replace $\mathbf{s}$ by $-\mathbf{s}$.
5. Output $(\rho, \mathbf{s})$.

The transitions up to Step 2 incur the same advantage losses as in the proof of Lemma 2. Here, besides the q'_{sign} evaluations corresponding to $\text{Hash}(\mu, \rho)$, each signing query makes two evaluations of H in $\text{Expand}_{\text{sk}}$ and one to compute μ . Key generation and final-forgery verification make four further evaluations in total, giving the term $3q_{\text{sign}} + 4$. Together with the existing constant term in Lemma 2, these evaluations yield the $3q_{\text{sign}} + 5$ term in the square root of the bound. The transition from Step 2 to Step 3, however, requires additional care because the QR-UOV signing algorithm imposes the rank restriction and obtains its target vector by applying RejSamp to a byte string.

Removing the rank restriction. We remove the rank restriction from the signing oracle of Step 2. Let $\text{Good}_{\text{rank}}$ be the event that all vectors $\mathbf{y}$ sampled during the signing queries in Step 2 without the rank restriction satisfy $\text{rank}(L_{\text{sk}}(\mathbf{y})) \geq m - \Delta$. Conditioned on $\text{Good}_{\text{rank}}$, the signing oracle of Step 2 without the rank restriction is identical to the one with the rank restriction. Moreover, by the definition of $\varepsilon_{\text{rank}}$ and the union bound,

$$\Pr[\text{Good}_{\text{rank}}] \geq 1 - q_{\text{sign}}\varepsilon_{\text{rank}}.$$

Since Step 2 with the rank restriction uses the signing oracle defined in Definition 3, its output is distributed as the output of Step 2 without the rank restriction, conditioned on $\text{Good}_{\text{rank}}$. Therefore,

$$\begin{aligned} & \Pr[\mathcal{A}_{\text{cma}} \text{ wins in Step 2 without the rank restriction}] \\ & \geq \Pr[\mathcal{A}_{\text{cma}} \text{ wins in Step 2 without the rank restriction} \mid \text{Good}_{\text{rank}}] \Pr[\text{Good}_{\text{rank}}] \\ & = \Pr[\mathcal{A}_{\text{cma}} \text{ wins in Step 2 with the rank restriction}] \Pr[\text{Good}_{\text{rank}}]. \end{aligned}$$

Consequently,

$$\begin{aligned} & \Pr[\mathcal{A}_{\text{cma}} \text{ wins in Step 2 with the rank restriction}] \\ & \leq \frac{\Pr[\mathcal{A}_{\text{cma}} \text{ wins in Step 2 without the rank restriction}]}{1 - q_{\text{sign}}\varepsilon_{\text{rank}}}. \end{aligned}$$

Signing-oracle simulation. We show that the signing oracle of Step 2 without the rank restriction and that of Step 3 induce the same signature distribution after applying the same canonicalization in both steps. To establish this, we separately consider the joint distribution of the preimage $\mathbf{s}$ and its image $\mathbf{t} = \mathcal{P}(\mathbf{s})$, and the conditional distribution of the byte string $\mathbf{r}$ given $(\mathbf{s}, \mathbf{t})$.

- We compare the distributions of $(\mathbf{s}, \mathbf{t})$ induced by Step 2 without the rank restriction with that induced by Step 3, applying the same canonicalization in both steps. As shown by Sakumoto et al. [SSH11], before canonicalization, the preimage $\mathbf{s}$ generated through retries in the modified UOV procedure follows the uniform distribution over $\mathbb{F}_q^n$. Hence, Step 3 samples $\mathbf{s}$ from the same distribution as Step 2, and $\mathbf{t} = \mathcal{P}(\mathbf{s})$ has the same distribution in both steps. In both steps, if $-\mathbf{s}$ is lexicographically smaller than $\mathbf{s}$, we replace $\mathbf{s}$ by $-\mathbf{s}$. Since the same deterministic canonicalization is applied to identically distributed preimages, the resulting distributions of $(\mathbf{s}, \mathbf{t})$ are also identical. Moreover, $\mathcal{P}(-\mathbf{s}) = \mathcal{P}(\mathbf{s})$, so canonicalization does not change $\mathbf{t}$.
- Conditioned on a fixed pair $(\mathbf{s}, \mathbf{t})$, we compare the distribution of $\mathbf{r}$ induced by the signing oracle of Step 2 without the rank restriction with that induced by Step 3. In Step 2, each byte string is sampled with the same probability. Moreover, for a fixed $\mathbf{t}$, the inversion test and the selection of the preimage $\mathbf{s}$ depend only on $\mathbf{t}$, and not on which element of $\mathcal{R}_{\mathbf{t}}$ produced it. Therefore, conditioned on $(\mathbf{s}, \mathbf{t})$, $\mathbf{r}$ is uniform over $\mathcal{R}_{\mathbf{t}}$. By

Lemma 7, $\text{InvRejSamp}(\mathbf{t}, \tau_m, m)$ also outputs a uniform element of $\mathcal{R}_{\mathbf{t}}$. Hence, conditioned on $(\mathbf{s}, \mathbf{t})$, $\mathbf{r}$ has the same distribution in Steps 2 and 3.

Hence, $(\mathbf{r}, \mathbf{s}, \mathbf{t})$ is distributed identically in Steps 2 and 3 without the rank restriction. The simulator uses this algorithm to obtain $\mathbf{r}$ in Step 3, thereby making the simulation of the signing oracle feasible in QR-UOV.

Reduction to INV. After the above modifications, the INV adversary $\mathcal{B}_{\text{inv}}$ can simulate the game of Step 3 as follows. Given the public key map $\mathcal{P}$, $\mathcal{B}_{\text{inv}}$ randomly selects seed_{pk} and provides $\text{pk} = (\text{seed}_{\text{pk}}, \{\bar{P}_{i,3}\}_{i \in [m]})$ to $\mathcal{A}_{\text{cma}}$. To ensure that the adversary can recover $\mathcal{P}$ from pk , for every $i \in [m]$, $\mathcal{B}_{\text{inv}}$ uses InvRejSamp to sample byte strings whose rejection-sampled values encode $\bar{P}_{i,1}$ and $\bar{P}_{i,2}$ in the prescribed order. It then programs the required prefixes of $H_{\text{prg}}(\text{seed}_{\text{pk}}, 2i - 1)$ and $H_{\text{prg}}(\text{seed}_{\text{pk}}, 2i)$ with these byte strings. Consequently, $\text{Expand}_{\text{pk}}(\text{seed}_{\text{pk}}, i)$ outputs $(\bar{P}_{i,1}, \bar{P}_{i,2})$ contained in $\mathcal{P}$. The bound of Lemma 2 therefore applies with the following adjustments. Since the same random oracle H models all uses of SHAKE256, the $3q_{\text{sign}} + 4$ additional evaluations of H are included in the first reprogramming term. The possible failures of RejSamp contribute the additive term $(q'_{\text{sign}} + 2m + 2)/2^\lambda$, while removing the rank restriction introduces the multiplicative factor $(1 - q_{\text{sign}}\varepsilon_{\text{rank}})^{-1}$ in the INV advantage. The programming of H_{prg} and the deterministic canonicalization of signatures introduce no additional loss. Combining these changes with Lemma 2 yields the bound stated in this lemma. $\square$

E.2 Proof of Lemma 5

Lemma 5 (restated). *For any quantum INV adversary $\mathcal{A}_{\text{inv}}$ against the trapdoor function $\mathbf{T}$ underlying QR-UOV, there exist adversaries $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ against the UOV problem with parameters $(q^\ell, v/\ell, m/\ell, m)$ and the QR-MQ problem with parameters $(q, (v+m), m, \ell)$, respectively, such that*

$$\text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{A}_{\text{inv}}) \leq \text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}) + \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}}),$$

where the running times of $\mathcal{B}_{\text{uov}}$ and $\mathcal{B}_{\text{mq}}$ are about that of $\mathcal{A}_{\text{inv}}$.

Proof. We prove the above statement by the following sequence of games.

1. Let Game_0 be the INV game played by $\mathcal{A}_{\text{inv}}$ against the trapdoor function of QR-UOV. By definition, $\Pr[\text{Game}_0() = 1] = \text{Adv}_{\mathbf{T}}^{\text{INV}}(\mathcal{A}_{\text{inv}})$.
2. Let Game_1 be the same as Game_0 except that the challenger chooses a random $\mathcal{P} \in \text{QR}_{q, (v+m), m, \ell}$ without the UOV structure. We can construct an adversary $\mathcal{B}_{\text{uov}}$ on the UOV problem with parameters $(q^\ell, v/\ell, m/\ell, m)$ as follows: when $\mathcal{B}_{\text{uov}}$ is given a multivariate quadratic map $\bar{\mathcal{P}}$ with $N = (v+m)/\ell$ variables and m equations over $\mathbb{F}_{q^\ell}$, it transforms $\bar{\mathcal{P}}$ into $\mathcal{P}$ with $(v+m)$ variables and m equations over $\mathbb{F}_q$ composed of m matrices in $W^{(N)}\mathcal{A}_f^{N,N}$. This transformation is performed by $W^{(N)}\phi$, where ϕ is

defined in Subsection 4.1. The adversary $\mathcal{B}_{\text{uov}}$ then runs $\mathcal{A}_{\text{inv}}$ on input $\mathcal{P}$. By Lemma 4, if $\mathcal{B}_{\text{uov}}$ is given a $\bar{\mathcal{P}} \in \text{UOV}_{q^\ell, v/\ell, m/\ell, m}$, then $\mathcal{B}_{\text{uov}}^{\mathcal{A}_{\text{inv}}}$ simulates Game_0 , and if $\mathcal{B}_{\text{uov}}$ is given a $\bar{\mathcal{P}} \in \text{MQ}_{q^\ell, N, m}$, then $\mathcal{B}_{\text{uov}}^{\mathcal{A}_{\text{inv}}}$ simulates Game_1 . Therefore, we have

$$|\Pr[\text{Game}_0() = 1] - \Pr[\text{Game}_1() = 1]| \leq \text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}).$$

3. From the construction, Game_1 is the QR-MQ game with parameters $(q, (v+m), m, \ell)$, and thus we have $\Pr[\text{Game}_1() = 1] = \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}})$.

Combining the game-transition bounds gives

$$\begin{aligned} & \left| \text{Adv}_{\text{T}}^{\text{INV}}(\mathcal{A}_{\text{inv}}) - \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}}) \right| \leq \text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}) \\ \Rightarrow & \text{Adv}_{\text{T}}^{\text{INV}}(\mathcal{A}_{\text{inv}}) \leq \text{Adv}_{q^\ell, v/\ell, m/\ell, m}^{\text{UOV}}(\mathcal{B}_{\text{uov}}) + \text{Adv}_{q, (v+m), m, \ell}^{\text{QR-MQ}}(\mathcal{B}_{\text{mq}}). \end{aligned}$$

□