SCIS 2025 2025 Symposium on
Cryptography and Information Security
Kokura, Japan, Jan. 28 – 31, 2025
The Institute of Electronics,
Information and Communication Engineers

On the Reference Implementation of QR-UOV and its Revised Version

Fumitaka Hoshino* Hiroki Furue† Yasuhiko Ikematsu‡ Tsuyoshi Takagi§
Haruhisa Kosuge† Kimihiro Yamakoshi† Rika Akiyama† Satoshi Nakamura†
Shingo Orihara† Koha Kinjo†

Abstract: Furue et al. proposed a post-quantum signature scheme called QR-UOV, and submitted it to the post-quantum cryptography standardization process for additional digital signature schemes. Recently NIST announced that 14 candidates, including QR-UOV, have advanced to the second round of the process, and encouraged the designers of QR-UOV to further optimize their implementation. In response to this encouragement, the designers decided to tweak the specification of QR-UOV and revise their implementation. In this work, we illustrate the details of this update and evaluate its effect.

Keywords: PQC, MPKCs, quotient ring UOV (QR-UOV), rejection sampling

1 Introduction

It used to be thought that, classical cryptography would not be broken by quantum computers in the foreseeable future, unless a tremendous technological breakthrough occurred. However, in recent years, there has been a huge investment in research into quantum computing, and it is often said that in the not-too-distant future, developments in quantum computing may threaten the security of classical cryptography. Accordingly, the Post-Quantum Cryptography (PQC) has suddenly become the mainstream of cryptographic research.

In December 2016, the National Institute of Standards and Technology (NIST) issued a public call for submissions to the PQC Standardization Process in response to the substantial development of quantum computing [16].

Multivariate public key cryptography (MPKC) based on the multivariate quadratic ($\mathcal{MQ}$) problem is considered a good candidate for PQC, because a decision variant of $\mathcal{MQ}$ problem is proven to be NP-complete [10] and thus schemes based on this problem are likely to be secure in the post-quantum era.

The unbalanced oil and vinegar signature scheme (UOV) [14], is a multivariate signature scheme proposed by Kipnis et al. at EUROCRYPT 1999, which has withstood various types of attacks for more than 20 years. UOV is a well-established signature scheme owing to its short signature and short execution time.

A multilayer UOV variant Rainbow [7] was selected as a third-round finalist in the NIST PQC project [15]. However, an attack on Rainbow proposed by Beullens at 2022 [3] broke the security of third round parameters and make the Rainbow scheme inefficient.

Subsequently, NIST announced the selection of the first algorithms to be standardized, however MPKC was never included there. Instead, NIST issued a call for additional PQC digital signature schemes in September 2022. In June 2023, NIST received 40 proper candidates, 10 of which were based on the $\mathcal{MQ}$ problem, most of them were variants of UOV.

Quotient ring UOV (QR-UOV) [8] is one of such UOV variants, whose public key size is relatively small, which was originally proposed by Furue et al. in ASIACRYPT 2021 [9]. A tweaked version was submitted to the NIST PQC Standardization Process later [8].

This submission includes some software code which naively implement the algorithms in the specification document [8]. It served to show that the scheme was "efficient", i.e., not infeasible. However, this implementation performed very poorly respect to the required time and memory to sign.

Recently NIST announced that 14 candidates, including QR-UOV, have advanced to the second round of the standardization process, and its internal report [1] states for QR-UOV that "NIST anticipates that the performance could be improved and encourages the designers to further optimize their implementation."

* Faculty of Information Systems, University of Nagasaki, 1-1-1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki, 851-2195, Japan. (hoshino@sun.ac.jp)
† NTT Social Informatics Laboratories, 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan. ({hiroki.furue, hrhs.kosuge, kimihiro.yamakoshi, rika.akiyama, satoshi.nakamura, shingo.orihara, kouha.kinjo}@ntt.com)
‡ Institute of Mathematics for Industry, Kyushu University, 744, Motooka, Nishi-ku, Fukuoka, 819-0395, Japan. (ikematsu@imi.kyushu-u.ac.jp)
§ Department of Mathematical Informatics, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. (takagi@g.ecc.u-tokyo.ac.jp)

In response to this encouragement, the designers decided to tweak the specification of QR-UOV and revise their implementation. In this work, we illustrate the details of this update and evaluate its effect.

1.1 Related Works

Various problems and improvements to QR-UOV have been published in many papers.

Hoshino et.al. suggested that the part of deriving F_1 and F_2 from $\mathsf{seed}_{\mathsf{sk}}$ and $\mathsf{seed}_{\mathsf{pk}}$ is the bottleneck of the signing algorithm. [11] As an improvement, they proposed the well-known technique of UOV which includes F_1 and F_2 in the secret key and skips the bottleneck part. This improvement increases the size of the secret key but achieves a significant speed-up. [11]

Amagasa et.al. indicated that the implementation is unlikely to benefit from vectorization by SIMD architecture due to the polynomial structure of each element in matrices. [2] To overcome this problem, they proposed a vectorized matrix operation of the extension field matrices, and apply to the key generation and signing algorithm of QR-UOV. They also identified that the matrix chain ordering in the signing algorithm of QR-UOV specification is not optimal, and specified an efficient order of matrix multiplication. [2]

Kannwischer et.al. pointed out the reference implementation of QR-UOV allocates huge arrays for signature generation. [13] This problem is caused by the order of sampling defined in the specification of QR-UOV. As a result of revising the specification, our new implementation does not have this problems.

2 Preliminary

2.1 Basic Notions and Notations

In this work, we identify the set of integers $\{0, 1\}$ with the set of truth values. Namely 0 is interpreted as false and 1 as truth. For a probabilistic Turing machine Y whose input is X and output is in $\{0, 1\}$, we say Y accepts X if $Y(X)$ returns 1, or Y rejects X if $Y(X)$ returns 0. The following list describes the notions and notations used in the following sections.

bit	one of the two symbols '0' or '1' .
bit string	an ordered sequence of bits.
octet	a bit string of length 8.
octet string	an ordered sequence of octets.
$\|\|$	a concatenation operator for two bit strings or for two octet strings.
$\mathbb{F}_q$	finite field with q elements for a prime power q.
$\lceil x \rceil$	for x a real number returns the smallest integer greater than or equal to x.
$\lfloor x \rfloor$	for x a real number returns the largest integer less than or equal to x.
$[n]$	for n a positive integer returns the set $\{1, \ldots, n\}$.

The following list is an overview of typical arrow expressions which we use to describe algorithms.

$X \leftarrow Y$	The value of the expression Y is assigned to the variable X.
$X \xleftarrow{\circ} Y$	A new value $X \circ Y$ is assigned to the variable X, for any binary operator $\circ$, e.g. $X \xleftarrow{\cup} Y$ means $X \leftarrow X \cup Y$.
$X \xleftarrow{\$} Y$	X is uniformly selected from Y, assuming that Y is a set.
$X \xleftarrow{\$} Y(\ldots)$	X is randomly selected from the output space of the probabilistic Turing machine Y according to the distribution of Y's output when Y's random tape is uniformly selected. If the Turing machine Y takes some input tapes, appropriate arguments are placed inside the parentheses.
$X \xmapsto{\$} Y$	A probabilistic Turing machine which takes X as an input tape and outputs Y.
$X \to Y$	All of maps from X to Y, assuming that X and Y are sets, which is identical with Y^X.
$X \mapsto Y$	the map which returns Y for X.

We also give some notations for representation matrices of elements of a quotient ring described in Subsection 2.3.

f	an irreducible polynomial in $\mathbb{F}_q[x]$ with degree ℓ.
Φ_g^f	an $\ell \times \ell$ matrix over $\mathbb{F}_q$ defined by equation (1) in Subsection 2.3.
$\mathcal{A}_f$	$\left\{\Phi_g^f \in \mathbb{F}_q^{\ell \times \ell} \;\middle|\; g \in \mathbb{F}_q[x]/(f)\right\}$.
W	an $\ell \times \ell$ matrix over $\mathbb{F}_q$ such that WX is symmetric for any $X \in \mathcal{A}_f$.
$W\mathcal{A}_f$	$\{WX \in \mathbb{F}_q^{\ell \times \ell} \mid X \in \mathcal{A}_f\}$.
$\mathcal{A}_f^{a,b}$	the set of $a\ell \times b\ell$ block matrices whose each component is an element of $\mathcal{A}_f$.
$W^{(a)}$	the $a\ell \times a\ell$ block diagonal matrix concatenating W diagonally a times.

The following is a list of the system parameter to specify the concrete instance of QR-UOV signature scheme.

ℓ, V, M	positive integers.
v	number of vinegar variables: $v = \ell \cdot V$.
m	number of oil variables (equals to # of equations): $m = \ell \cdot M$.
n	number of variables: $n = v + m$.
N	$N = V + M$.
λ	security parameter.

2.2 Signature Scheme

A digital signature scheme Σ consists of three probabilistic polynomial-time algorithms (`KeyGen`, `Sign`, `Verify`), which are defined as follows:

Key generation algorithm $\mathtt{KeyGen} : 1^\lambda \xmapsto{\$} (\mathtt{pk}, \mathtt{sk}) \in (\{0,1\}^*)^2$, returns a pair of public key $\mathtt{pk}$ and secret key $\mathtt{sk}$ for given security parameter 1^λ.

Signing algorithm $\mathtt{Sign} : \mathbf{M}, \mathtt{pk}, \mathtt{sk} \xmapsto{\$} \sigma \in \{0,1\}^*$, takes a message $\mathbf{M} \in \{0,1\}^*$ and the signer's key pair $(\mathtt{pk}, \mathtt{sk})$, and then generates a digital signature σ of the signer for the message $\mathbf{M}$.

Verification algorithm $\mathtt{Verify} : \mathbf{M}, \mathtt{pk}, \sigma \xmapsto{\$} \beta \in \{0, 1\}$, takes a message $\mathbf{M} \in \{0,1\}^*$, a public key $\mathtt{pk}$ and a signature σ, and accepts or rejects the message $\mathbf{M}$.

Let $\mathtt{Adv}^{\mathtt{complete}} : \mathbb{N} \to [0,1]$ be a function as follows.

$$\mathtt{Adv}^{\mathtt{complete}}(\lambda) := \Pr\left[\beta = 1 \;\middle|\; \begin{array}{l} (\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda), \\ \mathbf{M} \xleftarrow{\$} \{0,1\}^{\mathtt{poly}(\lambda)}, \\ \sigma \xleftarrow{\$} \mathtt{Sign}(\mathbf{M}, \mathtt{pk}, \mathtt{sk}), \\ \beta \xleftarrow{\$} \mathtt{Verify}(\mathbf{M}, \mathtt{pk}, \sigma). \end{array}\right].$$

We say a signature scheme $\Sigma := (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$ is complete in λ iff $\mathtt{Adv}^{\mathtt{complete}}(\lambda)$ is overwhelming in λ. In the following sections, we regard the completeness as a part of the syntax of signature scheme, thus we assume implicitly that any signature schemes are complete.

Let $\mathtt{invalid}$ be a set of messages and $\mathcal{O}_{\mathtt{pk},\mathtt{sk}}$ be a signing oracle defined as follows.

$\mathcal{O}_{\mathtt{pk},\mathtt{sk}}(\mathbf{M}) :=$

$\sigma \xleftarrow{\$} \mathtt{Sign}(\mathbf{M}, \mathtt{pk}, \mathtt{sk})$,
$\mathtt{invalid} \xleftarrow{\cup} \{\mathbf{M}\}$,
return σ.

Let $\mathcal{A}$ be a forger against the signature scheme Σ. We define $\mathtt{Adv}^{\mathtt{EUF\text{-}CMA}}_{\mathcal{A}} : \mathbb{N} \to [0,1]$ as

$$\mathtt{Adv}^{\mathtt{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda) := \Pr\left[\beta = 1 \;\middle|\; \begin{array}{l} (\mathtt{pk}, \mathtt{sk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda), \\ \mathtt{invalid} \leftarrow \varnothing, \\ (\mathbf{M}^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathtt{pk},\mathtt{sk}}}(\mathtt{pk}), \\ \beta \xleftarrow{\$} \mathtt{Verify}(\mathbf{M}^*, \mathtt{pk}, \sigma^*) \\ \qquad\qquad \wedge (\mathbf{M}^* \notin \mathtt{invalid}). \end{array}\right].$$

We say a signature scheme $\Sigma := (\mathtt{KeyGen}, \mathtt{Sign}, \mathtt{Verify})$ is EUF-CMA in λ iff for any probabilistic polynomial-time algorithm $\mathcal{A}$, $\mathtt{Adv}^{\mathtt{EUF\text{-}CMA}}_{\mathcal{A}}(\lambda)$ is negligible in λ.

2.3 Basic Concept of the QR-UOV

Let $f \in \mathbb{F}_q[x]$ be a polynomial in $\mathbb{F}_q$ of degree ℓ. For each element $g \in \mathbb{F}_q[x]/(f)$, there exists a matrix $\Phi^f_g \in \mathbb{F}_q^{\ell\times\ell}$ s.t.

$$(1, x, \ldots, x^{\ell-1})\Phi^f_g = (g, xg, \ldots, x^{\ell-1}g) \in (\mathbb{F}_q[x]/(f))^\ell. \tag{1}$$

Let $A_f := \{\Phi^f_g \in \mathbb{F}_q^{\ell\times\ell} \mid g \in \mathbb{F}_q[x]/(f)\}$. A_f is a subalgebra of $\mathbb{F}_q^{\ell\times\ell}$ because

$$\Phi^f_{g_1+g_2} = \Phi^f_{g_1} + \Phi^f_{g_2},$$
$$\Phi^f_{g_1 g_2} = \Phi^f_{g_1}\Phi^f_{g_2}.$$

Hence an element in A_f, i.e. $\ell\times\ell$ $\mathbb{F}_q$-matrix can be represented as an element in $\mathbb{F}_q[x]/(f)$, i.e. ℓ elements of $\mathbb{F}_q$. One might think that this property of A_f could be available to construct a UOV variant with short public key. However this technique cannot be applied directly to the UOV variants with another promising technique using symmetric matrices, since the elements of A_f are generally unstable under the transpose operation. Furue et al. solve this problem by introducing an invertible matrix $W \in \mathbb{F}_q^{\ell\times\ell}$ s.t. $W\Phi^f_g$ is transpose for any $\Phi^f_g \in A_f$, thus they propose a UOV variant using $WA_f := \{W\Phi^f_g \in \mathbb{F}_q^{\ell\times\ell} \mid \Phi^f_g \in A_f\}$, that is QR-UOV [9]. Therefore, throughout this work, we will identify the element $g \in \mathbb{F}_q[x]/(f)$ with its matrix form $W\Phi^f_g \in WA_f$. When we need to distinguish them, we write its matrix form $W\Phi^f_g$ as $\tilde{g}$ for $g \in \mathbb{F}_q[x]/(f)$, i.e.

$$\tilde{g} := W\Phi^f_g.$$

Moreover, for $G := (g_{ij})_{i\in[n_1],j\in[n_2]} \in (\mathbb{F}_q[x]/(f))^{n_1\times n_2}$, we define

$$\widetilde{G} := (W\Phi^f_{g_{ij}})_{i\in[n_1],j\in[n_2]} \in (WA_f)^{n_1\times n_2},$$

where $[n] := \{1, \ldots, n\}$. Similarly, for $m \in \mathbb{N}$, we denote the set of integers $\{1, \ldots, m\}$ as $[m]$ in the following sections.

3 The Round 1 QR-UOV

3.1 Key Generation

In this section, we describe the key generation algorithms of the Round 1 QR-UOV according to its specification document [8]. Let v be the number of vinegar variables, m be the number of oil variables which is equal to the number of equations, and $n = v + m$. From the notations in Subsection 2.1, the public and secret keys of QR-UOV are represented by elements of $\mathcal{A}_f^{N,N}$ and $W^{(N)}\mathcal{A}_f^{N,N} := \left\{W^{(N)} \cdot X \,|X \in \mathcal{A}_f^{N,N}\right\}$, where $N = n/\ell$ with the number n of variables.

The standard key generation of QR-UOV is described as follows:

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.
2. Choose an invertible matrix S from $\mathcal{A}_f^{N,N}$ randomly.
3. Compute the public key $P_i = S^\top F_i S$ $(i \in [m])$.

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 (Prop. 1 in [9]). *For $X \in \mathcal{A}_f^{N,N}$ and $Y \in W^{(N)}\mathcal{A}_f^{N,N}$, we have*

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

Subsequently, we apply an improved method restricting the secret key $\mathcal{S}$ to a specific compact form, which was first proposed by Czypek et al. [6]. Before describing the improved method, we prepare some notations: For the public key P_i ($i \in [m]$) and the secret key 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 suppose to limit the secret key S to the following compact form

$$S = \begin{pmatrix} I_v & S' \\ O & I_m \end{pmatrix}, \tag{2}$$

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

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

By using this equation, in the improved key generation step, $P_{i,1} \in W^{(V)}\mathcal{A}_f^{V,V}$, $P_{i,2} \in W^{(V)}\mathcal{A}_f^{V,M}$ ($i \in [m]$), and $S' \in \mathcal{A}_f^{V,M}$, where $V = v/\ell$ and $M = m/\ell$, are first generated from random seeds, and $P_{i,3} \in W^{(M)}\mathcal{A}_f^{M,M}$ ($i \in [m]$) is computed by

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

As a result, the public key is composed of $m \times m$ matrices $P_{i,3}$ ($i \in [m]$) and the 2λ-bit seed $\mathsf{seed_{pk}}$ for $P_{i,1}$, $P_{i,2}$ ($i \in [m]$), and the secret key is composed of the 2λ-bit seed $\mathsf{seed_{sk}}$ for S', where λ is the security parameter. The security of QR-UOV is not weakened by this optimization, since this does not affect the distribution of the public and secret keys. The resulting algorithm for key generation is shown in Algorithm 1.

Algorithm 1 The Round 1 KeyGen()

Input: parameters (q, v, m, ℓ), security parameter λ
Output: public key pk, secret key sk

1: $\mathsf{seed_{pk}}, \mathsf{seed_{sk}} \xleftarrow{\$} \{0,1\}^{2\lambda}$
2: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand_{pk}}(\mathsf{seed_{pk}})$
▷ $P_{i,1} \in W^{(V)}\mathcal{A}_f^{V,V}$ (symmetric), $P_{i,2} \in W^{(V)}\mathcal{A}_f^{V,M}$
3: $S' \leftarrow \mathsf{Expand_{sk}}(\mathsf{seed_{sk}})$ ▷ $S' \in A_f^{V,M}$
4: **for** i from 1 to m **do**
5: $\quad P_{i,3} \leftarrow -S'^\top P_{i,1} S' + P_{i,2}^\top S' + S'^\top P_{i,2}$
6: **end for**
7: **return** $(\mathsf{pk}, \mathsf{sk}) = \left(\left(\mathsf{seed_{pk}}, \{P_{i,3}\}_{i\in[m]}\right), \mathsf{seed_{sk}}\right)$

The Round 1 specification of QR-UOV [8] defined $\mathsf{Expand_{sk}}$ and $\mathsf{Expand_{pk}}$ as follows.

$\mathsf{Expand_{sk}}$ This expands the seed $\mathsf{seed_{sk}}$ for the secret key to $S' \in \mathcal{A}_f^{V,M}$. As we mentioned before, this S' can be represented as a $V \times M$ matrix over $\mathbb{F}_q[x]/(f)$. We sample the matrix in row-major order and sample each polynomial in $\mathbb{F}_q[x]/(f)$ in reverse degree order from the constant term to the coefficient of $x^{\ell-1}$.

$\mathsf{Expand_{pk}}$ This expands the seed $\mathsf{seed_{pk}}$ for the public key to $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]}$ where $P_{i,1}$ is a symmetric $v \times v$ matrix in $W^{(V)}\mathcal{A}_f^{V,V}$ and $P_{i,2}$ is a $v \times m$ matrix in $W^{(V)}\mathcal{A}_f^{V,M}$. Then, this $P_{i,1}$ and $P_{i,2}$ can be represented as $V \times V$ and $V \times M$ matrices over $\mathbb{F}_q[x]/(f)$. We here first sample $P_{1,1}, \ldots, P_{m,1}$ and then $P_{1,2}, \ldots, P_{m,2}$. For each matrix, we sample in row-major order and sample each polynomial in $\mathbb{F}_q[x]/(f)$ in reverse degree order. Note that for $P_{i,1}$ we sample only the upper-triangular elements due to the symmetry.

3.2 Signature Generation

The signature generation of QR-UOV is mainly depending on 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

$$S^{-1} = \begin{pmatrix} I_v & -S' \\ O & I_m \end{pmatrix},$$

from equation (2). We here add a modification for the EUF-CMA security proof proposed by Sakumoto et al. [17]. See Algorithm 2 for more details.

We here describe the inversion of the central map $\mathcal{F}$ in the signature generation. We first choose values for the vinegar variables $y_1, \ldots, y_v$ randomly. We then choose λ-bit random salt r and compute $\mathbf{t} \in \mathbb{F}_q^m$ by applying a hash function Hash on the input concatenating a given message $\mathbf{M}$ and the salt r, namely $\mathbf{t} := \mathsf{Hash}(\mathbf{M}||r)$. If the linear system for the oil variables $x_{v+1}, \ldots, x_n$

$$\mathcal{F}(y_1, \ldots, y_v, x_{v+1}, \ldots, x_n) = \mathbf{t}, \tag{4}$$

has solutions, then we obtain the signature by applying $\mathcal{S}^{-1}$ into $(y_1, \ldots, y_v,\ y_{v+1}, \ldots, y_n)$, where $(y_{v+1}, \ldots, y_n)$ is a randomly chosen solution of equation (4). If there exists no solution of equation (4), then we choose a new salt and update $\mathbf{t}$ until equation (4) has solutions.

The main difference from the standard signature generation algorithm is that if equation (4) has no solution, then we choose a new random salt instead of choosing new vinegar variables. By doing so, the signature $\mathbf{s}$ satisfying $\mathcal{P}(\mathbf{s}) = \mathsf{Hash}(\mathbf{M}||r)$ is uniformly distributed in $\mathbb{F}_q^n$, and this fact enables us to prove

the EUF-CMA security of QR-UOV [8]. For the efficiency, we confirm that the expected number of computing $\mathbf{t} = \mathsf{Hash}(\mathbf{M}||r)$ until equation (4) has solutions is approximately 2.0 for any parameter sets by assuming that equation (4) is a randomized system for x_{v+1}, $\ldots, x_n$.

Algorithm 2 The Round 1 Sign($\mathbf{M}, \mathsf{pk}, \mathsf{sk}$)

Input: message $\mathbf{M}$, public key pk, secret key sk
Output: signature σ

$\left(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}\right) \leftarrow \mathsf{pk}$
$\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{sk}$
$\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
$S' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$
for i from 1 to m **do**
  $F_{i,1} \leftarrow P_{i,1}$
  $F_{i,2} \leftarrow -P_{i,1}S' + P_{i,2}$
end for
$S \leftarrow \begin{pmatrix} I_v & S' \\ 0_{m\times v} & I_m \end{pmatrix}$
$\mathbf{y} = (y_1, \ldots, y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$
$L \leftarrow \begin{pmatrix} 2\mathbf{y}^\top F_{1,2} \\ \vdots \\ 2\mathbf{y}^\top F_{m,2} \end{pmatrix}$ ▷ $L \in \mathbb{F}_q^{m\times m}$
$\mathbf{u} \leftarrow \left(\mathbf{y}^\top F_{1,1}\mathbf{y}, \ldots, \mathbf{y}^\top F_{m,1}\mathbf{y}\right)^\top$ ▷ $\mathbf{u} \in \mathbb{F}_q^m$
repeat
  $r \xleftarrow{\$} \{0,1\}^\lambda$
  $\mathbf{t} \leftarrow \mathsf{Hash}(\mathbf{M}||r)$ ▷ $\mathbf{t} \in \mathbb{F}_q^m$
until $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has solutions for $\mathbf{x}$.
Choose one solution $(y_{v+1}, \ldots, y_n) \in \mathbb{F}_q^m$ of $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ randomly.
$\mathbf{s} \leftarrow S^{-1}(y_1, \ldots, y_v, y_{v+1}, \ldots, y_n)^\top$
return $\sigma = (r, \mathbf{s})$

3.3 Pseudo-Random $\mathbb{F}_q$ Sampler

Key generation and signing algorithms of QR-UOV sample many $\mathbb{F}_q$ (pseudo-)random elements to generate its public-key and signature. When q is prime, an element of $\mathbb{F}_q \cong \mathbb{Z}/q\mathbb{Z}$ is typically represented by an integer in $\{0, \ldots, q-1\}$. While the output of a pseudo-random number generator(PRNG) is typically in $\{0, 1\}^*$. Since the QR-UOV specification requires that q is odd, we must convert a bit string into an integer in $\{0, \ldots, q-1\}$.

If uniformity of random elements in $\mathbb{F}_q$ is not required, it is sufficient to sample a bit string of suitable length, interpret it as an integer, divide by q, and take the remainder. Such an $\mathbb{F}_q$ sampler is very fast, however it is not preferable for security proof, because it distorts the distribution of the $\mathcal{MQ}$ problem on which QR-UOV is based.

Therefore QR-UOV employed the rejection sampling, which is well known as an exact sampling method. The rejection sampling is a simple method of generating random numbers with an arbitrary efficiently computable probability distribution, from an uniform random number generator, which was published by John von Neumann in the early 1950s [12]. Especially in our cases, the logic of rejection is quite simple, since the distribution is just a step function

$$\begin{array}{rcccl} p & : & \{0, \ldots, 2^{\lceil \log_2 q \rceil}\} & \to & [0,1], \\ & & x & \mapsto & \delta(x < q)/q, \end{array}$$

where

$$\begin{array}{rcccl} \delta & : & \{\top, \bot\} & \to & \{0,1\}, \\ & & \top & \mapsto & 1, \\ & & \bot & \mapsto & 0. \end{array}$$

The Round 1 specification specified its pseudo-random $\mathbb{F}_q$ generator as follows. [8]

> Many random finite field elements are used when generating keys and sampling vinegar variables. For example, since an element of $\mathbb{F}_q[x]/(f)$ can be represented as a matrix $\mathbb{F}_q^{\ell\times\ell}$, the Expand functions in the KeyGen algorithm would generate elements of the finite field $\mathbb{F}_q$. A random bit sequence is generated using a hash function and retrieved for every $\lceil \log_2 q \rceil$ bit to generate these elements. By dividing a hash value into $\lceil \log_2 q \rceil$ bits, a sequence of random numbers in the range of $[0, 2^{\lceil \log_2 q \rceil})$ can be obtained. Even so, in the range of $[0, 2^{\lceil \log_2 q \rceil})$, q is the only number that does not belong to $\mathbb{F}_q$. Therefore, when q is obtained from the sequence of random numbers, q should be skipped and not chosen. Also, when obtaining the element of $\mathbb{F}_q^m$, the first m numbers that are non q values should be selected. This method, called rejection sampling, ...

4 Update Plans for Round 2

No attacks have found that threaten QR-UOV since its submission to NIST. Therefore, we have no plans to change the essential parts of the scheme or update the parameters from Round 1. Consideration was given to whether or not the technique proposed by Sakumoto et al. [17] could be omitted to improve performance, however, we decided to keep it.

4.1 Update for Key Generation

In round 2 we will update the specification of `KeyGen`() as follows.

- In the Round 1 specification, the length of each seed has 2λ bits. Since it was considered too long for the security parameter, the length of each seed will be set to λ bits in round 2.
- Round 1 rejection sampler has poor performance. Moreover, it is difficult to support for various techniques due to the unpredictability of rejection. Therefore we have a plan to update our rejection sampler. The following are the considerations for this update.
  - support for low memory device:
    The Round 1 specification of sampling $P_{1,1}, \ldots, P_{m,1}$ followed by sampling $P_{1,2}, \ldots, P_{m,2}$ in $\mathsf{Expand}_{\mathsf{pk}}$ was very problematic because Round 1 rejection $\mathbb{F}_q$ sampler can't sample the subsequent elements until the preceding sample is finished due to the unpredictability of rejection. Therefore to sample $P_{1,2}$, all of $P_{1,1}, \ldots, P_{m,1}$ must be sampled in advance. The problem noted by Kannwischer et al. [13] is caused by this specification of $\mathsf{Expand}_{\mathsf{pk}}$. Therefore we have a plan to update $\mathsf{Expand}_{\mathsf{pk}}$ et.al.

- support for concurrent/parallel sampling:
  In UOV-type signatures, operations on matrices such as $\{P_{i,1}\}_{i\in[m]}$ and $\{P_{i,2}\}_{i\in[m]}$ can be performed independently for each i. Therefore, the matrix operations for each i can be performed in parallel. We will introduce support for this technique in this update.
- support for lazy sampling:
  Within the signing and verification algorithms, the matrices $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]}$ are sampled pseudo-randomly by calling $\mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$, then producted with the vector $\mathbf{s}$ in the signature. The vector $\mathbf{s}$ can be expected to have zeros in its elements with probability $1/q$. Therefore it can be expected that some rows of matrices do not need to be sampled with probability $1/q$. Sampling to achieve this speed-up by manipulating PRNG states is called lazy sampling. [4] Due to differences in sampling methods, the benefits of this technique is limited in QR-UOV. Therefore, it was decided not to introduce support for this technique in this update.

In round 2 we will employ the following $\mathsf{Expand}_{\mathsf{pk}}$ to address the drawback of the Round 1 specification.

$\mathsf{Expand}_{\mathsf{pk}}$ This expands the seed $\mathsf{seed}_{\mathsf{pk}}$ for the public key 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 $\mathsf{Expand}_{\mathsf{pk}}$ 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 $\mathsf{RejSampPRG}$, $(v_1,\ldots,v_{n_1}) \in \mathbb{F}_q^{n_1}$ with $n_1 := \ell V(V+1)/2$ and $(v'_1,\ldots,v'_{n_2}) \in \mathbb{F}_q^{n_2}$ with $n_2 := \ell VM$ are expanded from $\mathsf{seed}_{\mathsf{pk}}$. Then, $\mathsf{ExpandSymmetricMatrixVxV}$ and $\mathsf{ExpandMatrixVxM}$ 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 3 shows the details.

Algorithm 3 The Round 2 $\mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}}, i)$

Input: a public seed $\mathsf{seed}_{\mathsf{pk}}$ and counter i
Output: matrices $(\bar{P}_{i,1}, \bar{P}_{i,2}) \in (\mathbb{F}_{q^\ell})^{V\times V} \times (\mathbb{F}_{q^\ell})^{V\times M}$

$n_1 := \ell V(V+1)/2$
$(v_1,\ldots,v_{n_1}) \leftarrow \mathsf{RejSampPRG}(\mathsf{seed}_{\mathsf{pk}}, 2i-1, \tau_1, n_1)$ ▷ $\tau_1 = \tau_{q,\lambda}(n_1)$
$\bar{P}_{i,1} \leftarrow \mathsf{ExpandSymmetricMatrixVxV}(v_1,\ldots,v_{n_1})$
$n_2 := \ell VM$
$(v'_1,\ldots,v'_{n_2}) \leftarrow \mathsf{RejSampPRG}(\mathsf{seed}_{\mathsf{pk}}, 2i, \tau_2, n_2)$ ▷ $\tau_2 = \tau_{q,\lambda}(n_2)$
$\bar{P}_{i,2} \leftarrow \mathsf{ExpandMatrixVxM}(v'_1,\ldots,v'_{n_2})$
return $(\bar{P}_{i,1}, \bar{P}_{i,2})$

The resulting key generation algorithm is given in Algorithm 5

4.2 Update for Signature Generation

In round 2 we will update the implementation of Signature Generation as follows.

- In Round 1, the signing algorithm was specified to take the digest of the original message and salt in the repeat loop. Because the calculation to digest the same message with different salts may be performed many times, when signing a very long message, efficient implementations have to calculate the message digest halfway through before looping, and save the state of the calculation to be used over and over again. In Round 2 specification, QR-UOV will employ the BUFF transform [5] which does not directly sign the original message but rather signs a message representative that is obtained by hashing of public key and message. BUFFing also helps to improve the security proof.
- The Round 2 implementation will employ the optimized matrix chain ordering of Amagasa et.al. [2], and some of Round 2 implementation will employ their vectorized matrix operation method. [2],

The resulting algorithm for signature generation is shown in Algorithm 4.

Algorithm 4 The Round 2 $\mathrm{Sign}(\mathbf{M}, \mathsf{pk}, \mathsf{sk})$

Input: message $\mathbf{M} \in \mathbb{B}^*$, public key pk, secret key sk
Output: signature σ

$\left(\mathsf{seed}_{\mathsf{pk}}, \{\bar{P}_{i,3}\}_{i\in[m]}\right) \leftarrow \mathsf{pk}$
$\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{sk}$
$\bar{S}' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$
$\mathbf{y} = (y_1,\ldots,y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$
for i from 1 to m **do**
  $(\bar{P}_{i,1}, \bar{P}_{i,2}) \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}}, i)$
  $\mathbf{y'}_i^\top \leftarrow \mathbf{y}^\top W^{(N)} \phi^{-1}\left(\bar{P}_{i,1}\right)$
  $\mathbf{L}_i^\top \leftarrow 2\left(-\mathbf{y'}_i^\top W^{(N)} \phi^{-1}\left(\bar{S}'\right) + \mathbf{y}^\top W^{(N)} \phi^{-1}\left(\bar{P}_{i,2}\right)\right)$
  $u_i \leftarrow \mathbf{y'}_i^\top \mathbf{y}$
end for
$L \leftarrow \left(\mathbf{L}_1^\top,\ldots,\mathbf{L}_m^\top\right)^\top$ ▷ $L \in \mathbb{F}_q^{m\times m}$
$\mathbf{u} \leftarrow (u_1,\ldots,u_m)^\top$ ▷ $\mathbf{u} \in \mathbb{F}_q^m$
$\mu \leftarrow \mathsf{SHAKE256}(\mathsf{seed}_{\mathsf{pk}} \| \mathrm{BytesToBits}(\mathbf{M}), 512)$
repeat
  $r \xleftarrow{\$} \{0,1\}^\lambda$
  $\mathbf{t} \leftarrow \mathsf{Hash}(\mu, r)$ ▷ $\mathbf{t} \in \mathbb{F}_q^m$
until $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has solutions for $\mathbf{x}$.
Choose one solution $(y_{v+1},\ldots,y_n)^\top \in \mathbb{F}_q^m$ of $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ randomly.
$\mathbf{s} \leftarrow (y_1,\ldots,y_v, y_{v+1},\ldots,y_n)^\top + \left((y_1,\ldots,y_v)\cdot\phi^{-1}\left(\bar{S}'\right)^\top \| (0,\ldots,0)\right)^\top$ ▷ $\mathbf{s} \in \mathbb{F}_q^n$
return $\sigma = (r, \mathbf{s})$

4.3 Update for Pseudo-Random $\mathbb{F}_q$ Sampler

In the Round 1 implementation, the random $\mathbb{F}_q$ sampler first reads $\lceil \log_2 q \rceil$ bits random number $x \in \{0, \ldots, 2^{\lceil \log_2 q \rceil}\}$ from the beginning of the unread random tape. Then it outputs x if $x < q$, otherwise it repeats the same thing, consuming the next $\lceil \log_2 q \rceil$ bits.

Algorithm 5 The Round 2 KeyGen()

Output: public key pk, secret key sk

1: $\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}} \xleftarrow{\$} \{0,1\}^{\lambda}$
2: $\bar{S}' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$ ▷ $\bar{S}' \in \mathbb{F}_{q^\ell}^{V \times M}$
3: **for** i from 1 to m **do**
4: $(\bar{P}_{i,1}, \bar{P}_{i,2}) \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}}, i)$
▷ $\bar{P}_{i,1} \in \mathbb{F}_{q^\ell}^{V \times V}$ (symmetric), $\bar{P}_{i,2} \in \mathbb{F}_{q^\ell}^{V \times M}$
5: $\bar{P}_{i,3} \leftarrow -\bar{S}'^{\top} \bar{P}_{i,1} \bar{S}' + \bar{P}_{i,2}^{\top} \bar{S}' + \bar{S}'^{\top} \bar{P}_{i,2}$
6: **end for**
7: **return** $(\mathsf{pk}, \mathsf{sk}) = \left(\left(\mathsf{seed}_{\mathsf{pk}}, \{\bar{P}_{i,3}\}_{i \in [m]}\right), \mathsf{seed}_{\mathsf{sk}}\right)$

The disadvantage of this method is the following threefold.

- There is no guarantee of the required random tape length.
- Since this method is bit-oriented, it requires a lot of shift and logic operations in the software implementation, which is inefficient.
- This method packs and arranges the accepted input, so that most bytes of the input random tape are moved from their own address to other addresses. Hence, bus usage tends to be high.

To overcome the first drawback, we will modify the rejection sampling to allow negligible error probability. If negligible error probability is allowed, length of the random tape can be guaranteed. Given λ, q, n, let $\tau_{q,\lambda}(n)$ be the number of elements in $\{0, \ldots, 2^{\lceil \log_2 q \rceil} - 1\}$ required to generate n random elements over $\mathbb{F}_q$ with success probability $1 - 2^{-\lambda}$. It is derived by

$$\tau_{q,\lambda}(n) := \min\{t \in \mathbb{N} \mid P(n, t, q/2^{\lceil \log_2 q \rceil}) \leq 2^{-\lambda}\}.$$

$P(n,t,p)$ 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 well known in statistics, and many numerical packages provide functions to compute it. Although some upper bounds for τ may be derived using something like Chernoff bounds, $\tau_{q,\lambda}(n)$ can be directly evaluated by combining such a numerical function with some root-finding algorithm. An efficient root-finding scheme like the Newton's method should be employed to evaluate it on the fly, however the bisection method is sufficient for just obtaining pre-calculated values. Figure 1 is a plot of τ-n ratio where $\lambda = 256$. If n is sufficiently large, t/n approaches 1, so the increase in random tape length consumed by rejection sampling is insignificant.

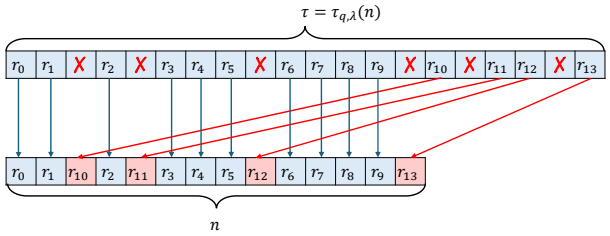

Fig 2: reuse of input tape by overwriting rejected bytes

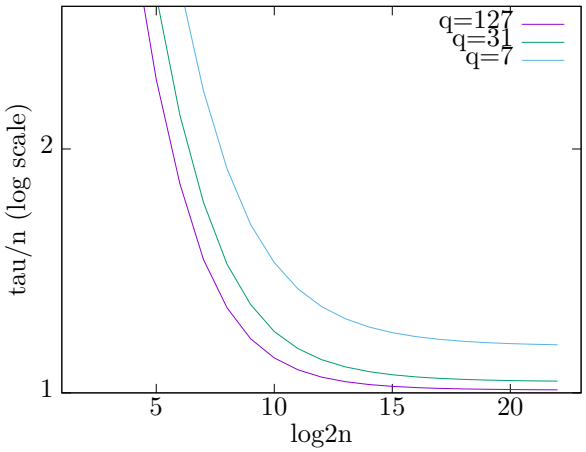

Fig 1: $\tau_{q,\lambda}(n)/n$ vs $\log_2 n$ where $\lambda = 256$.

To address the second drawback, we will choose byte-oriented sampling, which simply discard unnecessary bits. The third drawback can be solved by overwriting only the input bytes that should be rejected and reusing the input tape as its output. Using this method, the sampler can divert the input tape directly to the output, thus reducing bus usage. Figure 2 illustrates this method.

5 Performance Evaluation

We plan that AES counter mode is available as well as SHAKE as a pseudo-random number generator in the Round 2 specification. Table 1 show the timing data of new implementations using AES option. For the reader's convenience, we refer to the timing data of the Round 1 implementations from its specification document [8] in Table 2. These implementations are written in C and do not use special processor instructions, but they ignore the 32-bit environment. The experimental environment is as follows, which is exactly the same as the environment in the Round 1 specification document. [8] It shows that the new signing algorithm is 7-8 times faster than the Round 1 version.

Processor: AMD EPYC 7763.
Clock Speed: Boost Clock : Up to 3.5GHz, Base Clock: 2.45GHz.
Memory: 128GB (32GB RDIMM, 3200MT/s, Dual Rank, 8Gb base x4)

Operating System: Linux 5.19.0-41-generic, gcc version 11.3.0.
Measurement Software: supercop-20221122.

Table 1: the New Implementation (Mcycles)

category	(q, v, m, ℓ)	keygen	sign	verify
I	$(127, 156, 54, 3)$	10.061	1.920	1.589
	$(31, 165, 60, 3)$	13.977	2.657	2.284
	$(31, 600, 70, 10)$	45.296	12.993	11.222
	$(7, 740, 100, 10)$	112.478	32.046	28.808
III	$(127, 228, 78, 3)$	43.802	5.904	5.123
	$(31, 246, 87, 3)$	59.691	8.533	7.308
	$(31, 890, 100, 10)$	197.123	39.703	34.319
	$(7, 1100, 140, 10)$	477.225	102.542	92.883
V	$(127, 306, 105, 3)$	126.797	14.051	12.138
	$(31, 324, 114, 3)$	166.735	17.671	15.464
	$(31, 1120, 120, 10)$	443.360	74.691	64.839
	$(7, 1490, 190, 10)$	1574.162	253.770	223.180

Table 2: the Round 1 Implementation (Mcycles)

category	(q, v, m, ℓ)	keygen	sign	verify
I	$(127, 156, 54, 3)$	16.700	13.419	10.575
	$(31, 165, 60, 3)$	20.223	15.813	11.614
	$(31, 600, 70, 10)$	93.984	92.480	73.814
	$(7, 740, 100, 10)$	177.911	167.711	99.755
III	$(127, 228, 78, 3)$	65.263	52.290	37.159
	$(31, 246, 87, 3)$	85.616	65.286	42.450
	$(31, 890, 100, 10)$	387.796	362.721	245.240
	$(7, 1100, 140, 10)$	905.595	822.727	385.265
V	$(127, 306, 105, 3)$	217.373	158.856	81.309
	$(31, 324, 114, 3)$	233.036	168.576	87.673
	$(31, 1120, 120, 10)$	826.049	783.495	474.469
	$(7, 1490, 190, 10)$	2528.767	2220.364	844.445

References

[1] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J.M. Kelsey, J. Lichtinger, C.A. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. SmithTone, N. Waller, and Y.K. Liu, "Status report on the first round of the additional digital signature schemes for the nist post-quantum cryptography standardization process," 2024-10-24 04:10:00 2024. doi:https://doi.org/10.6028/NIST.IR.8528.

[2] H. Amagasa, R. Ueno, and N. Homma, "Performance Improvement of QR-UOV Software Based on SIMD Operations." In *Proc. of SCIS 2024, 2024 Symposium on Cryptography and Information Security Nagasaki, Japan, Jan. 23 - 26, 2024*. IEICE, 2024.

[3] W. Beullens, "Breaking rainbow takes a weekend on a laptop," Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II, ed. Y. Dodis and T. Shrimpton, Lecture Notes in Computer Science, vol.13508, pp.464–479, Springer, 2022. doi:10.1007/978-3-031-15979-4_16.

[4] W. Beullens, M.S. Chen, J. Ding, B. Gong, M.J. Kannwischer, J. Patarin, B.Y. Peng, D. Schmidt, C.J. Shih, C. Tao, and B.Y. Yang, "UOV: Unbalanced Oil and Vinegar." https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/UOV-spec-web.pdf.

[5] C. Cremers, S. Düzlü, R. Fiedler, M. Fischlin, and C. Janson, "Buffing signature schemes beyond unforgeability and the case of post-quantum signatures," 2021 IEEE Symposium on Security and Privacy (SP), pp.1696–1714, 2021. doi:10.1109/SP40001.2021.00093.

[6] P. Czypek, S. Heyse, and E. Thomae, "Efficient implementations of MQPKS on constrained devices," Cryptographic Hardware and Embedded Systems – CHES 2012, ed. E. Prouff and P. Schaumont, Berlin, Heidelberg, pp.374–389, Springer Berlin Heidelberg, 2012.

[7] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings, ed. J. Ioannidis, A.D. Keromytis, and M. Yung, Lecture Notes in Computer Science, vol.3531, pp.164–175, 2005. doi:10.1007/11496137_12.

[8] H. Furue, Y. Ikematsu, F. Hoshino, T. Takagi, K. Yasuda, T. Miyazawa, T. Saito, and A. Nagai, "QR-UOV." https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/qruov-spec-web.pdf.

[9] H. Furue, Y. Ikematsu, Y. Kiyomura, and T. Takagi, "A new variant of unbalanced oil and vinegar using quotient ring: QR-UOV," Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV, ed. M. Tibouchi and H. Wang, Lecture Notes in Computer Science, vol.13093, pp.187–217, Springer, 2021. doi:10.1007/978-3-030-92068-5_7.

[10] M.R. Garey and D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., USA, 1990.

[11] F. Hoshino, H. Furue, Y. Ikematsu, T. Takagi, K. Yasuda, T. Miyazawa, A. Nagai, R. Akiyama, and K. Kinjo, "More Efficient Software Implementation of QR-UOV." In *Proc. of SCIS 2024, 2024 Symposium on Cryptography and Information Security Nagasaki, Japan, Jan. 23 - 26, 2024*. IEICE, 2024.

[12] V.N. J., "Various Techniques used in Connection with Random Digits," National Bureau of Standards Series, vol.12, pp.36–38, 1951. URL: https://cir.nii.ac.jp/crid/1572261550886416128.

[13] M.J. Kannwischer, M. Krausz, R. Petri, and S.Y. Yang, "pqm4: Benchmarking NIST additional post-quantum signature schemes on microcontrollers." Cryptology ePrint Archive, Paper 2024/112, 2024. URL: https://eprint.iacr.org/2024/112.

[14] A. Kipnis, J. Patarin, and L. Goubin, "Unbalanced oil and vinegar signature schemes," Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding, ed. J. Stern, Lecture Notes in Computer Science, vol.1592, pp.206–222, Springer, 1999. doi:10.1007/3-540-48910-X_15.

[15] "Recommendation for key management, special publication 800-57 part 1, NIST, 03/2007," 2007.

[16] "NIST: Post-quantum cryptography CSRC.." https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization.

[17] K. Sakumoto, T. Shirai, and H. Hiwatari, "On provable security of UOV and HFE signature schemes against chosen-message attack," Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings, ed. B. Yang, Lecture Notes in Computer Science, vol.7071, pp.68–82, Springer, 2011. doi:10.1007/978-3-642-25405-5_5.