SCIS 2024 2024 Symposium on Cryptography and Information Security
Nagasaki, Japan, Jan. 23 – 26, 2024
The Institute of Electronics, Information and Communication Engineers

More Efficient Software Implementation of QR-UOV

Fumitaka Hoshino* Hiroki Furue† Yasuhiko Ikematsu‡ Tsuyoshi Takagi†
Kan Yasuda§ Toshiyuki Miyazawa¶ Akira Nagai¶ Rika Akiyama¶ 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 of NIST in response to call for additional digital signature schemes. This submission includes some software code which naively implements the algorithms in the specification document. This implementation has extremely compact form of signing key, which slow down the signing process instead. While it is preferable for the secret to be as compact as possible, redundant forms of signing keys can greatly speed up the signing process. There is a trade-off between the compactness of signing key and the speed of the signing process. For applications where signing speed is critical, it may not be a problem if the signing key is somewhat redundant. In this work, we investigate this trade-off, implement QR-UOV with a redundant key form, and evaluate its performance on some x86 environments.

Keywords: PQC, MPKCs, quotient ring UOV (QR-UOV), software implementation

1 Introduction

In December 2016, the National Institute of Standards and Technology (NIST) issued a public call for submissions to the Post-Quantum Cryptography (PQC) Standardization Process in response to the substantial development of quantum computing [11].

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 [7] 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) [9], 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 [3] was selected as a third-round finalist in the NIST PQC project [10]. However, an attack on Rainbow proposed by Beullens at 2022 [1] broke the security of third round parameters and make the Rainbow scheme inefficient.

* Faculty of Information Systems, University of Nagasaki, 1-1-1, Manabino, Nagayo-cho, Nishisonogi-gun, Nagasaki, 851-2195, Japan. (hoshino@sun.ac.jp)
† Department of Mathematical Informatics, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. ({furue-hiroki261,takagi}@g.ecc.u-tokyo.ac.jp)
‡ Institute of Mathematics for Industry, Kyushu University, 744, Motooka, Nishi-ku, Fukuoka, 819-0395, Japan. (ikematsu@imi.kyushu-u.ac.jp)
§ NTT Social Informatics Laboratories, 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan. ({kan.yasuda, toshiyuki.miyazawa, akira.nagai, rika.akiyama, kouha.kinjo}@ntt.com)

Subsequently, NIST announced the selection of the first algorithms to be standardized, including the three digital signature schemes: CRYSTALS-Dilithium, FALCON, and SPHINCS+, 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) [5] is one of such UOV variants, whose public key size is relatively small, which was originally proposed by Furue et al. in ASIACRYPT 2021 [6]. A tweaked version was submitted to the NIST PQC Standardization Process later [5]. This submission includes some software code which naively implements the algorithms in the specification document [5].

This implementation has extremely compact form of signing key whose length is only 64 $\sim$ 128 bytes, however it slow down the signing process instead. While it is preferable for the secret to be as compact as possible, redundant forms of signing keys can greatly speed up the signing process. In fact, other UOV variants have such a key form [12].

There is a trade-off between the compactness of signing key and the speed of the signing process. For applications where signing speed is critical, it may not be a problem if the signing key is somewhat redundant. In this work, we investigate this trade-off, implement QR-UOV with a redundant key form, and evaluate its performance on some x86 environments.

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 $\{1,\dots,n\}$.

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.
$X \overset{\circ}{\leftarrow} Y$	A new value $X \circ Y$ is assigned to the variable X, for any binary operator $\circ$, e.g. $X \overset{\cup}{\leftarrow} Y$ means $X \leftarrow X \cup Y$.
$X \overset{\$}{\leftarrow} Y$	X is uniformly selected from Y, assuming that Y is a set.
$X \overset{\$}{\leftarrow} Y(\dots)$	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.
$X \overset{\$}{\mapsto} Y$	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.
$\mathcal{A}_f$	$\left\{\Phi_g^f \in \mathbb{F}_q^{\ell\times\ell} \mid g \in \mathbb{F}_q[x]/(f)\right\}$.
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 `KeyGen` : $1^\lambda \overset{\$}{\mapsto} (\mathtt{pk}, \mathtt{sk}) \in (\{0,1\}^*)^2$, returns a pair of public key `pk` and secret key `sk` for given security parameter 1^λ.

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

Verification algorithm `Verify` : $\mathbf{M}, \mathtt{pk}, \sigma \overset{\$}{\mapsto} \beta \in \{0,1\}$, takes a message $\mathbf{M} \in \{0,1\}^*$, a public key `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.

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

We say a signature scheme $\Sigma :=$ (`KeyGen`, `Sign`, `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 `invalid` be a set of messages and $\mathcal{O}_{\mathtt{pk},\mathtt{sk}}$ be a signing oracle defined as follows.

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

$\sigma \xleftarrow{\$} \texttt{Sign}(\mathbf{M}, \texttt{pk}, \texttt{sk})$,

$\texttt{invalid} \xleftarrow{\cup} \{\mathbf{M}\}$,

return σ.

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

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

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

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

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 [6]. 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 Algorithms of QR-UOV [5]

In this section, we describe the algorithms of QR-UOV signature scheme according to its specification document [5].

3.1 Key Generation

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 [6])**.** *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. [2]. 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

$$P_i = \begin{pmatrix} P_{i,1} & P_{i,2} \\ P_{i,2}^\top & P_{i,3} \end{pmatrix}, \quad F_i = \begin{pmatrix} F_{i,1} & F_{i,2} \\ F_{i,2}^\top & 0_{m\times m} \end{pmatrix},$$

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

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

Algorithm 1 KeyGen()

```
Input: parameters $(q, v, m, \ell)$, security parameter $\lambda$
Output: public key $\mathsf{pk}$, secret key $\mathsf{sk}$
1: $\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}} \xleftarrow{\$} \{0,1\}^{2\lambda}$
2: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
                              $\triangleright$ $P_{i,1} \in W^{(V)}\mathcal{A}_f^{V,V}$ (symmetric), $P_{i,2} \in W^{(V)}\mathcal{A}_f^{V,M}$
3: $S' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$          $\triangleright$ $S' \in A_f^{V,M}$
4: for $i$ from 1 to $m$ do
5:     $P_{i,3} \leftarrow -S'^{\top} P_{i,1} S' + P_{i,2}^{\top} S' + S'^{\top} P_{i,2}$
6: end for
7: return $(\mathsf{pk}, \mathsf{sk}) = \left(\left(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}\right), \mathsf{seed}_{\mathsf{sk}}\right)$
```

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 $\mathsf{seed}_{\mathsf{pk}}$ for $P_{i,1}$, $P_{i,2}$ $(i \in [m])$, and the secret key is composed of the 2λ-bit seed $\mathsf{seed}_{\mathsf{sk}}$ 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.

We here use the following two functions $\mathsf{Expand}_{\mathsf{sk}}$ and $\mathsf{Expand}_{\mathsf{pk}}$ to expand the public and secret keys from randomly chosen seeds

$\mathsf{Expand}_{\mathsf{sk}}$ This expands the seed $\mathsf{seed}_{\mathsf{sk}}$ 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 $x^{\ell-1}$.

$\mathsf{Expand}_{\mathsf{pk}}$ This expands the seed $\mathsf{seed}_{\mathsf{pk}}$ 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. [13]. 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 [5]. 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 $x_{v+1}, \ldots, x_n$.

3.3 Signature Verification

The signature verification of QR-UOV is the same as that of the plain UOV. Given the public key pk, a message $\mathbf{M}$, and a signature $\sigma = (r, \mathbf{s})$, the authenticity of the signature is checked as follows:

- Use the hash function Hash to compute $\mathbf{t} = \mathsf{Hash}(\mathbf{M}||r)$.
- Compute $\mathbf{t}' \in \mathbb{F}_q^m$ by substituting the signature $\mathbf{s} \in \mathbb{F}_q^n$ for 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. See Algorithm 3 for more details.

4 QR-UOV with redundant key

As in the previous section, QR-UOV employed a compact key form introduced by Czypek et al. [2, 5]. We would like to relax this compaction with respect to

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

```
Input: message $\mathbf{M}$, public key pk, secret key sk
Output: signature $\sigma$
 1: $(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}) \leftarrow \mathsf{pk}$
 2: $\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{sk}$
 3: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
 4: $S' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$
 5: for $i$ from 1 to $m$ do
 6:     $F_{i,1} \leftarrow P_{i,1}$
 7:     $F_{i,2} \leftarrow -P_{i,1}S' + P_{i,2}$
 8: end for
 9: $S \leftarrow \begin{pmatrix} I_v & S' \\ 0_{m\times v} & I_m \end{pmatrix}$
10: $\mathbf{y} = (y_1, \dots, y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$
11: $L \leftarrow \begin{pmatrix} 2\mathbf{y}^\top F_{1,2} \\ \vdots \\ 2\mathbf{y}^\top F_{m,2} \end{pmatrix}$    ▷ $L \in \mathbb{F}_q^{m\times m}$
12: $\mathbf{u} \leftarrow (\mathbf{y}^\top F_{1,1}\mathbf{y}, \dots, \mathbf{y}^\top F_{m,1}\mathbf{y})^\top$    ▷ $\mathbf{u} \in \mathbb{F}_q^m$
13: repeat
14:     $r \xleftarrow{\$} \{0,1\}^\lambda$
15:     $\mathbf{t} \leftarrow \mathsf{Hash}(\mathbf{M}||r)$    ▷ $\mathbf{t} \in \mathbb{F}_q^m$
16: until $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has solutions for $\mathbf{x}$.
17: Choose one solution $(y_{v+1}, \dots, y_n) \in \mathbb{F}_q^m$ of $L\mathbf{x} =$
    $\mathbf{t} - \mathbf{u}$ randomly.
18: $\mathbf{s} \leftarrow S^{-1}(y_1, \dots, y_v, y_{v+1}, \dots, y_n)^\top$
19: return $\sigma = (r, \mathbf{s})$
```

signing keys in this section. Algorithm 4 is a relaxed version of Algorithm 1, which returns

$$\mathsf{sk}' = \left(S', \{F_{i,1}\}_{i\in[m]}, \{F_{i,2}\}_{i\in[m]}\right),$$

instead of $\mathsf{sk} = \mathsf{seed}_{\mathsf{sk}}$. Algorithm 5 is the corresponding signing algorithm.

The computational cost of Algorithm 4 is almost the same as that of Algorithm 1. However the size of sk' is much greater than the size of sk.

The actual signing key for Algorithm 2 is a concatenation of $\mathsf{seed}_{\mathsf{sk}}$ and $\mathsf{seed}_{\mathsf{pk}}$, requiring only 4λ bits in total, while the signing key of Algorithm 5 is a concatenation of $S' \in (\mathbb{F}_q^\ell)^{V\times M}$, $\{F_{i,1}\}_{i\in[m]} \in (\mathbb{F}_q^\ell)^{V\times V\times m}$, and $\{F_{i,2}\}_{i\in[m]} \in (\mathbb{F}_q^\ell)^{V\times M\times m}$, which requirs at least

$$\lceil m \cdot v \cdot (1 + m + (v+\ell)/2) \cdot \ell^{-1} \cdot \log_2 q \rceil$$

bits in total. The sizes of signing keys for concrete parameters are listed in Table 1. In fact, the actual memory size for the signing key of Algorithm 5 can be much larger than that due to alignment and other effects.

On the other hand, the computational cost of Algorithm 5 is much smaller than that of Algorithm 2.

Columns of "sign2" in table 1 show the performance results for two software implementations of the Algorithm 5 written in C for 64-bit environments. One uses

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

```
Input: message $\mathbf{M}$, public key pk, signature $\sigma$
Output: accept or reject
 1: $(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}) \leftarrow \mathsf{pk}$
 2: $(r, \mathbf{s}) \leftarrow \sigma$
 3: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
 4: for $i$ from 1 to $m$ do
 5:     $P_i \leftarrow \begin{pmatrix} P_{i,1} & P_{i,2} \\ P_{i,2}^\top & P_{i,3} \end{pmatrix}$
 6: end for
 7: $\mathbf{t} \leftarrow \mathsf{Hash}(\mathbf{M}||r)$
 8: $\mathbf{t}' \leftarrow (\mathbf{s}^\top P_1 \mathbf{s}, \dots, \mathbf{s}^\top P_m \mathbf{s})^\top$
 9: return accept if $\mathbf{t} = \mathbf{t}'$ and reject otherwise.
```

Algorithm 4 KeyGen2()

```
Input: parameters $(q, v, m, \ell)$, security parameter $\lambda$
Output: public key pk, secret key sk'
 1: $\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}} \xleftarrow{\$} \{0,1\}^{2\lambda}$
 2: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
        ▷ $P_{i,1} \in W^{(V)}\mathcal{A}_f^{V,V}$ (symmetric),
    $P_{i,2} \in W^{(V)}\mathcal{A}_f^{V,M}$
 3: $S' \leftarrow \mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$    ▷ $S' \in A_f^{V,M}$
 4: for $i$ from 1 to $m$ do
 5:     $F_{i,1} \leftarrow P_{i,1}$
 6:     $F_{i,2} \leftarrow -P_{i,1}S' + P_{i,2}$
 7:     $P_{i,3} \leftarrow S'^\top F_{i,2} + P_{i,2}^\top S'$
 8: end for
 9: return $(\mathsf{pk}, \mathsf{sk}') =$
10:     $\left((\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}), (S', \{F_{i,1}\}_{i\in[m]}, \{F_{i,2}\}_{i\in[m]})\right)$
```

avx2 intrinsics and the other does not use special processor instructions. For the reader's convenience, we refer to the timing data of keygen, sign and verify from the specification document [5]. The experimental environment is as follows, which is exactly the same as the environment in the specification document [5] except for stacksize limit.

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.
Stacksize Limit: unlimited.

For more comparison, we refer to the timing data of another UOV variant called VOX from its specification document [12], shown in table 2. Similar results are obtained despite the fact that the scheme, implementation and experimental environment are all different. See [12] for more details on VOX.

Table 1: Timing data and sizes of signing keys of QR-UOV (Mcycles)

category	(q, v, m, ℓ)	keygen	sign	sign2 (avx2)	sign2 (C)	verify	4λ (bits)	$\lvert sk' \rvert$ (bits)
I	$(127, 156, 54, 3)$	16.700	13.419	0.614	0.856	10.575	512	2639459
	$(31, 165, 60, 3)$	20.223	15.813	0.748	1.117	11.614	512	2370583
	$(31, 600, 70, 10)$	93.984	92.480	9.070	6.808	73.814	512	7823667
	$(7, 740, 100, 10)$	177.911	167.711	11.473	11.813	99.755	512	9888627
III	$(127, 228, 78, 3)$	65.263	52.290	1.664	2.542	37.159	768	8057926
	$(31, 246, 87, 3)$	85.616	65.286	2.227	3.429	42.450	768	7510438
	$(31, 890, 100, 10)$	387.796	362.721	26.084	20.001	245.240	768	24294884
	$(7, 1100, 140, 10)$	905.595	822.727	39.958	36.687	385.265	768	30090353
V	$(127, 306, 105, 3)$	217.373	158.856	5.543	6.464	81.309	1024	19498116
	$(31, 324, 114, 3)$	233.036	168.576	5.001	7.556	87.673	1024	16987405
	$(31, 1120, 120, 10)$	826.049	783.495	49.494	36.816	474.469	1024	45676898
	$(7, 1490, 190, 10)$	2528.767	2220.364	89.971	84.278	844.445	1024	74787121

Algorithm 5 Sign2($\mathbf{M}$, sk$'$)

```
Input: message $\mathbf{M}$, secret key sk$'$
Output: signature $\sigma$
1: $(S', \{F_{i,1}\}_{i\in[m]}, \{F_{i,2}\}_{i\in[m]}) \leftarrow \mathsf{sk}'$
2: $S \leftarrow \begin{pmatrix} I_v & S' \\ 0_{m\times v} & I_m \end{pmatrix}$
3: $\mathbf{y} = (y_1, \dots, y_v)^\top \xleftarrow{\$} \mathbb{F}_q^v$
4: $L \leftarrow \begin{pmatrix} 2\mathbf{y}^\top F_{1,2} \\ \vdots \\ 2\mathbf{y}^\top F_{m,2} \end{pmatrix}$    $\triangleright\ L \in \mathbb{F}_q^{m\times m}$
5: $\mathbf{u} \leftarrow \left(\mathbf{y}^\top F_{1,1}\mathbf{y}, \dots, \mathbf{y}^\top F_{m,1}\mathbf{y}\right)^\top$    $\triangleright\ \mathbf{u} \in \mathbb{F}_q^m$
6: repeat
7:     $r \xleftarrow{\$} \{0,1\}^\lambda$
8:     $\mathbf{t} \leftarrow \mathsf{Hash}(\mathbf{M}||r)$    $\triangleright\ \mathbf{t} \in \mathbb{F}_q^m$
9: until $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ has solutions for $\mathbf{x}$.
10: Choose one solution $(y_{v+1}, \dots, y_n) \in \mathbb{F}_q^m$ of $L\mathbf{x} = \mathbf{t} - \mathbf{u}$ randomly.
11: $\mathbf{s} \leftarrow S^{-1}(y_1, \dots, y_v, y_{v+1}, \dots, y_n)^\top$
12: return $\sigma = (r, \mathbf{s})$
```

Table 2: Timing data of VOX (cached SK) (Mcycles)

category	(q, o, v, t, c)	Sign (avx2)	Sign (C)
I	$(251, 8, 9, 6, 6)$	0.488	0.517
III	$(1021, 10, 11, 7, 7)$	1.887	2.061
V	$(4093, 12, 13, 8, 8)$	8.902	9.772

References

[1] 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.

[2] 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.

[3] 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.

[4] H. Furue, Y. Ikematsu, F. Hoshino, Y. Kiyomura, T. Saito, and T. Takagi, "Secure Parameters for Multivariate Polynomial Signature Scheme QR-UOV." In *Proc. of SCIS 2023, 2023 Symposium on Cryptography and Information Security Fukuoka, Japan, Jan. 24 - 27, 2023.* IEICE, 2023.

[5] 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.

[6] 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.

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

[8] F. Hoshino, H. Furue, Y. Ikematsu, T. Saito, Y. Kiyomura, and T. Takagi, "Efficient Software Implementation of Signature Scheme QR-UOV." In *Proc. of SCIS 2023, 2023 Symposium on Cryptography and Information Security Fukuoka, Japan, Jan. 24 - 27, 2023.* IEICE, 2023.

[9] 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.

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

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

[12] J. Patarin, B. Cogliati, J.C. Faugère, P.A. Fouque, L. Goubin, R. Larrieu, G. Macario-Rat, and B. Minaud, "Vox Specification v1.0 - 06/01/2023." https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/vox-spec-web.pdf.

[13] 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.