TUOV: Triangular Unbalanced Oil and Vinegar

Algorithm Specifications and Supporting Documentation

Version 1.0

Principal Submitter:
Jintai Ding
jintai.ding@gmail.com

Auxiliary Submitters:
Boru Gong, Hao Guo, Xiaoou He, Yi Jin,
Yuansheng Pan, Dieter Schmidt, Chengdong Tao,
Danli Xie, Bo-Yin Yang, Ziyu Zhao

May 30, 2023

tuovsig@gmail.com
https://www.tuovsig.org

Contents

1 Introduction 3

2 Preliminaries 5
2.1 Notations 5
2.2 Conventions 5

3 Specifications 6
3.1 Design Rationale 6
3.2 TUOV Scheme Description 11
3.3 Recommended Parameter Sets 18
3.4 Choice of Symmetric Primitives 18

4 Security Analysis 20
4.1 Security Reductions 20
4.2 Attack Scenarios 22

5 Implementations and Performance 26
5.1 Reference Implementation 26
5.2 AVX2 Optimized Implemention 29

6 Summary: Strengths and Limitations of TUOV 31

References 32

1 Introduction

This document presents **T**riangular **U**nbalanced **O**il and **V**inegar (TUOV), a digital signature scheme derived from UOV.

The Unbalanced Oil and Vinegar (UOV) digital signature scheme was first proposed in 1999 [1]. This cryptographic mechanism leverages the hash-and-sign paradigm underpinned by a trapdoored multivariate quadratic map. Over the course of two decades, UOV has withstood rigorous cryptanalytic scrutiny, which validates its robust security and consistent reliability. However, it shares a common feature with numerous other Multivariate Public Key Cryptosystems (MPKC); that is, it lacks a security proof grounded in weaker, more favorable assumptions.

Our TUOV scheme aims to correlate its security to a problem which can eventually be reduced to a heuristically challenging problem. Specifically, a Multivariate Quadratic (MQ) problem can be methodically reduced to align with the problem our TUOV scheme addresses.

The TUOV scheme demonstrates superior performance in terms of time efficiency and signature size. Especially when it comes to the NIST security level 1, TUOV shows several advantages over other post-quantum digital signature alternatives such as Dilithium[2], Falcon[3], and SPHINCS+[4].

- **Signature size**. TUOV signature can be as short as 80 bytes, are more compact, with notably smaller lengths compared to the other PQC candidates .
- **Signing speed**. TUOV showcases an impressive speed in generating signatures, significantly outpacing the competitors.
- **Verification speed**. TUOV equates the verification speed of Dilithium and notably surpasses Falcon and SPHINCS+, displaying a significant advantage in this aspect.

Table 1: Recommended parameter sets and and the corresponding key/signature sizes for TUOV variants.

	NIST Security Level		$\lvert\mathsf{upk}\rvert$	$\lvert\mathsf{usk}\rvert$	$\lvert\mathsf{cpk}\rvert$	$\lvert\mathsf{csk}\rvert$	$\lvert\sigma\rvert$
		(n, m, m_1, q)	(bytes)	(bytes)	(bytes)	(bytes)	(bytes)
`tuov-Ip`	1	(112, 44, 22, 256)	278 432	239 391	42 608	48	112
`tuov-Is`	1	(160, 64, 32, 16)	412 160	350 272	65 552	48	80
`tuov-III`	3	(184, 72, 36, 256)	1 225 440	1 048 279	186 640	48	184
`tuov-V`	5	(244, 96, 48, 256)	2 869 440	2 443 711	442 384	48	244

Beyond these characteristics, TUOV is admired for its simplicity, making it easy and straightforward to implement. Its streamlined structure reduces the risk of implementation errors, making it an extremely practical option for real-world usage.

In essence, TUOV offers certain confidence in its security beyond any other MPKCs, and competitive performance against the new NIST standards in most respects.

§2: Preliminaries. We introduce notations and digital signature conventions for completeness.

§3: Specifications. Section 3 starts with a brief introduction to UOV signature scheme and TUOV design rationale. Then we specify three variants of TUOV which offer various tradeoffs between space efficiency and time efficiency, so as to accommodate a variety of different use-cases. We conclude by proposing four sets of recommended parameters summarized in Table 1, as well as the choice of symmetric primitives.

§4: Security Analysis. In Section 4 we adopt a two-pronged approach. We first delve into the EUF-CMA security of TUOV. Our proof sketch provisionally maps the scheme's security to the TUOV problem, which can be proven to be at least as hard as the MQ problem. Secondly, we conduct an extensive strength analysis, considering major threats to MPKCs, such as direct, Kipnis-Shamir, Intersection, and MinRank attacks. We aim to ensure the robustness of our recommended parameter sets against these attacks, thereby achieving the NIST-prescribed security level, respectively.

§5: Implementations and Performance. To fully demonstrate the strengths of TUOV in practice, we describe in Section 5 the implementations of TUOV over NIST PQC Referenced Platform, together with the experimental results. Please refer to [5] for the full details on our implementations.

§6: Advantages and Limitations. The advantages and limitations of TUOV are summarized in Section 6.

2 Preliminaries

2.1 Notations

κ	the security parameter
$\mathbb{F}_q$	finite field with q elements
$\mathbf{A}, \mathbf{P}, \mathbf{Q}$	boldface capital letters denote matrices over $\mathbb{F}_q$
$\mathbf{a}, \mathbf{b}, \ldots$	lowercase letters in boldface denote column vectors over $\mathbb{F}_q$
$\mathbf{x} = (x_1, \ldots, x_n)^\top$	column vector by default
$[n]$	the set $\{1, 2, \ldots, n\}$
$\mathbf{0}_k$	the k-dimensional zero vector $\begin{bmatrix} 0 & \cdots & 0 \end{bmatrix}^\top$
$A \mathrel{+}= B$	evaluate A as $A + B$
Setup	system parameter setup algorithm
KeyGen	key generation algorithm
Sign	signing algorithm
Verify	verification algorithm
UT	for a square matrix $\mathbf{M}$, $\mathsf{UT}(\mathbf{M})$ denotes the unique upper triangular matrix $\mathbf{M}'$ such that the difference $\mathbf{M}' - \mathbf{M}$ is skew-symmetric

2.2 Conventions

Generally speaking, a digital signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of three probabilistic polynomial-time algorithms where KeyGen is the key generation algorithm, Sign is the signing algorithm and Verify is the verification algorithm.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$. pk is a public key and sk is the associated secret key.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)$. σ is a signature of the message $\mu \in \{0,1\}^*$.
- $b := \mathsf{Verify}(\mathsf{pk}, \mu, \sigma)$. It outputs $b \in \{\mathsf{accept}, \mathsf{reject}\}$, suggesting whether it accepts the signature σ as a valid signature on μ for the public key pk (*i.e.*, $b = \mathsf{accept}$) or not (*i.e.*, $b = \mathsf{reject}$).

In addition, a digital signature Π may be endowed with another probabilistic polynomial-time algorithm Setup running as $\mathsf{params} \leftarrow \mathsf{Setup}(1^\kappa)$.

We say a digital signature scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is *correct*, if for any sufficiently large κ, and any $\mu \in \{0,1\}^*$, it holds

$$\Pr[\mathsf{Verify}(\mathsf{pk}, \mu, \mathsf{Sign}(\mathsf{sk}, \mu)) = 1] = 1 - \mathsf{negl}(\kappa);$$

where the probability is taken over the randomness of the key generation and signing algorithms, and $\mathsf{negl}(\kappa)$ denotes a function that is negligible in the security parameter κ. Moreover, the *standard* security definition for a digital signature scheme is that it should be existentially unforgeable under chosen-message attack, or of EUF-CMA security for short. Please refer to [6] for formal security definitions.

3 Specifications

In this section we present the design of the TUOV digital signature scheme in full detail. Our specification is organized as follows. First, we describe the design rationale, i.e., the triangular map, behind TUOV. Then, in Section 3.2, we specify three variants of TUOV. The TUOV digital signature scheme is explained as a collection of five interrelated algorithms. These consist of the usual key generation, signing, and verification algorithms, supplemented by a secret key expansion algorithm and a public key expansion algorithm. This strategic design allows the key generation algorithm to produce compact forms of a secret key and its associated public key, which can then be expanded by the appropriate algorithms for use in the signing or verification processes. In Section 3.3 we propose four sets of recommended parameters. We conclude the section by specifying our recommended choice of symmetric primitives in Section 3.4.

3.1 Design Rationale

To facilitate our forthcoming discussion, we first provide a block-wise representation of the quadratic polynomial's coefficient matrix. Then we revisit the structure of the UOV scheme. Following this, we unveil a general TUOV scheme, outlining its underlying design rationale. It is worth noting that, the TUOV scheme to be proposed in Section 3.2 exemplifies a specifically parameterized variant of the general TUOV, offering superior signing efficiency.

3.1.1 Polynomials

Given parameters n, m, d, q. A quadratic polynomial f over $\mathbb{F}_q$ in n variables $x_1, \dots, x_n$ has the following form

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij} \cdot x_i x_j + \sum_{i=1}^{n} \beta_i \cdot x_i + \gamma,$$

and it has unique representations as

$$\begin{aligned}
f(\mathbf{x}) &= \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + \gamma \\
&= \begin{bmatrix} \mathbf{x}^{(1)\top} & \mathbf{x}^{(2)\top} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{A}^{(4)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{(1)\top} & \mathbf{b}^{(2)\top} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \end{bmatrix} + \gamma \\
&= \begin{bmatrix} \mathbf{x}^{(1)\top} & \mathbf{x}^{(2d1)\top} & \mathbf{x}^{(2d2)\top} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2d1)} & \mathbf{A}^{(2d2)} \\ \mathbf{0}_{d\times(n-m)} & \mathbf{A}^{(4d1)} & \mathbf{A}^{(4d2)} \\ \mathbf{0}_{(m-d)\times(n-m)} & \mathbf{0}_{(m-d)\times d} & \mathbf{A}^{(4d4)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2d1)} \\ \mathbf{x}^{(2d2)} \end{bmatrix} \\
&\quad + \begin{bmatrix} \mathbf{b}^{(1)\top} & \mathbf{b}^{(2d1)\top} & \mathbf{b}^{(2d2)\top} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2d1)} \\ \mathbf{x}^{(2d2)} \end{bmatrix} + \gamma
\end{aligned}$$

where $n \geq m \geq d \geq 1$, $\mathbf{A} \in \mathbb{F}_q^{n\times n}$, $\mathbf{A}^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$, $\mathbf{A}^{(4)} \in \mathbb{F}_q^{m\times m}$, $\mathbf{A}^{(4d1)} \in \mathbb{F}_q^{d\times d}$ and $\mathbf{A}^{(4d4)} \in \mathbb{F}_q^{(m-d)\times(m-d)}$ are upper-triangular matrices, every matrix(vector) with superscript (1) has $n-m$ columns (resp. rows), with superscript (2) has m columns(*resp.*rows), with superscript ($d1$) has d columns (*resp.*rows) and with superscript ($d2$) has $m-d$ columns (*resp.*rows).

Definition 1. An (n, m)**-OV-polynomial** f over $\mathbb{F}_q$ is defined as

$$\sum_{i=1}^{n-m}\sum_{j=1}^{n} \alpha_{i,j} \cdot x_i x_j + \sum_{i=1}^{n} \beta_i \cdot x_i + \gamma$$

and has unique representation as

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{0}_{m\times m} \end{bmatrix} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + \gamma$$

where $\mathbf{A}^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ is upper-triangular matrix, $\mathbf{A}^{(2)} \in \mathbb{F}_q^{(n-m)\times m}$, $\mathbf{b} \in \mathbb{F}_q^n$ is column vector and $\mathbf{x} = (x_1, \dots, x_n)^\top$ is column vector of variables.

Definition 2. And for $d \geq 1$, an (n, m, d)**-TOV-polynomial** f over $\mathbb{F}_q$ is defined as

$$\sum_{i=n-m+1}^{n-m+d} \sum_{j=n-m+1}^{n-m+d} \alpha_{i,j}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n-m} \sum_{j=1}^{n} \alpha_{i,j}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} \cdot x_i + \gamma^{(k)}$$

and has unique notation as

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2d1)} & \mathbf{A}^{(2d2)} \\ \mathbf{0}_{d\times(n-m)} & \mathbf{A}^{(4d1)} & \mathbf{0}_{d\times(m-d)} \\ \mathbf{0}_{(m-d)\times(n-m)} & \mathbf{0}_{(m-d)\times d} & \mathbf{0}_{(m-d)\times(m-d)} \end{bmatrix} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + \gamma$$

where $\mathbf{A}^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$ and $\mathbf{A}^{(4d1)} \in \mathbb{F}_q^{d\times d}$ are upper-triangular matrices, $\mathbf{A}^{(2d1)} \in \mathbb{F}_q^{(n-m)\times d}$, $\mathbf{A}^{(2d2)} \in \mathbb{F}_q^{(n-m)\times(m-d)}$, and $\mathbf{b} \in \mathbb{F}_q^n$ is column vector.

Definition 3. A *UOV central map* in relation to $\mathsf{params} = (n, m, q)$ is $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m, \mathbf{x} \mapsto \mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top$ where f_k's are (n, m)-OV polynomials over $\mathbb{F}_q$. And a *UOV map* in relation to $\mathsf{params} = (n, m, q)$ is $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$, where $\mathcal{F}$ is a UOV central map in relation to params and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is an affine invertible transformation.

Definition 4. A *TUOV central map* in relation to $\mathsf{params} = (n, m, m_1, m_2, q)$ is $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m, \mathbf{x} \mapsto \mathcal{F}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top$, where

$$f_k(\mathbf{x}) \text{ is } \begin{cases} (n, m_1)\text{-OV-polynomial}, & k = 1, \dots, m_1 \\ (n, m, k - m_1)\text{-TOV-polynomial}, & k = m_1 + 1, \dots, m_2 \\ (n, m - m_2 + m_1 - 1)\text{-OV-polynomial}, & k = m_2 + 1, \dots, m. \end{cases}$$

And a *TUOV map* in relation to $\mathsf{params} = (n, m, m_1, m_2, q)$ is $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ where $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ are invertible affine transformations and $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a TUOV central map in relation to params.

3.1.2 Brief on UOV

History of UOV. The origins of UOV scheme, along with its variants, could be traced back to Patarin's groundbreaking linearization equations attack [7] in 1995 against the Matsumoto-Imai cryptosystem. Following this innovative stride, Patarin ingeniously converted the attack's underlying principle into the design of the Oil and Vinegar signature scheme (OV) [1] in 1997. However, when an invariant subspace attack [8] cracked the *balanced* version of this scheme in 1998, Kipnis, Patarin, and Goubin promptly introduced the Unbalanced Oil and Vinegar (UOV) digital signature scheme [9] in the following year, 1999. The hallmark simplicity ingrained in UOV's design, coupled with the impressive resilience it has shown against any discovered vulnerabilities over two decades of relentless cryptanalysis, furnishing a compelling argument for the scheme's enduring security.

UOV key pair. Given security parameter κ and $\mathsf{params} = (n, m, q) \leftarrow \mathsf{Setup}(1^\kappa)$, UOV scheme has key pair $(\mathsf{pk} = \mathcal{P}, \mathsf{sk} = (\mathcal{F}, \mathcal{T}))$. In the pair, *UOV map* $\mathcal{P} = \mathcal{F} \circ \mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is the public key. *UOV central map* $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m, \mathbf{x} \mapsto \begin{bmatrix} f_1(\mathbf{x}) & \cdots f_m(\mathbf{x}) \end{bmatrix}^\top$ with f_k an

```
**KeyGen**(params = $(n, m, q)$):
    Choose OV-polynomials $f^{(1)}(x_1, ..., x_n), ..., f^{(m)}(x_1, ..., x_n)$ uniformly at random
    $\mathcal{F} := (f^{(1)}, ..., f^{(m)})$
    Choose an affine invertible map $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ uniformly at random
    $\mathcal{P} := \mathcal{F} \circ \mathcal{T}$
    pk := $\mathcal{P}$
    sk := $(\mathcal{F}, \mathcal{T})$
    **return** (pk, sk)

**Sign**(params, sk = $(\mathcal{F}, \mathcal{T}), \mu \in \{0,1\}^*$):
    $\mathbf{t} \leftarrow \mathsf{Hash}(\mu)$
    **repeat**
        $\mathbf{v} \leftarrow \mathbb{F}_q^{n-m}$
    **until** $\Delta_{\mathbf{t}} := \left\{ \begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \in \mathbb{F}_q^n \middle| \mathcal{F}\left( \begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} \right) = \mathbf{t} \right\} \neq \varnothing$
    $\mathbf{w} \leftarrow \Delta_{\mathbf{t}}$
    **return** $\sigma := \mathbf{w}$

**Verify**(params, pk = $\mathcal{P}, (\mu, \sigma = \mathbf{w})$):
    $\mathbf{t} \leftarrow \mathsf{Hash}(\mu)$
    $\mathbf{t}' := \mathcal{P}(\mathbf{w})$
    **return** $(\mathbf{t} == \mathbf{t}')$
```

Figure 1: The key generation, signing and verification algorithms of UOV.

(n, m)-OV polynomial over the field $\mathbb{F}_q$ for every $k \in [m]$. Additionally, $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is defined as an invertible affine transformation. It's important to note that, with the OV structure, for every $\mathbf{x}^{(1)} \in \mathbb{F}_q^{n-m}$, $f_k(\mathbf{x})$ is linear in $\mathbf{x}^{(2)}$, and consequently, $\mathcal{F}$ can be efficiently inverted. This feature underpins the process of the signing algorithm.

3.1.3 Blueprint of TUOV

Triangular map. The **T**riangular in the name TUOV refers to a triangular map (or, *de Jonquiére* map) as $\mathcal{J} : \mathbb{F}_q^n \to \mathbb{F}_q^n, \mathbf{x} \mapsto (x_1, x_2 + g_2(x_1), \cdots, x_n + g_n(x_1, \ldots, x_{n-1}))^\top$ where g_i is a polynomial over $\mathbb{F}_q$ in $i-1$ variables. Notice that given any vector $\mathbf{a} = (a_1, \ldots, a_n)^\top \in \mathbb{F}_q^n$, it is easy to find a pre-image $\mathbf{u} \in \mathbb{F}_q^n$ under $\mathcal{J}$ as we have

$$u_i = \begin{cases} a_i & i = 1 \\ a_i - g_i(u_1, \ldots, u_{i-1}) & i \in [n] \setminus \{1\} \end{cases}.$$

Hence the triangular map $\mathcal{J}$ is efficiently invertible.

Triangular variables in TUOV. The TUOV scheme, which is basically a variation of the UOV scheme, uses a similar strategy by first determining the vinegar values and then using them to obtain the oils. However, the TUOV scheme introduces some *triangular vinegar* variables through the use of a triangular map. Once the original vinegars are determined, these triangular vinegars are evaluated using the inverse of the triangular map. Then, both types of vinegars - the original and the triangular - are used together for oil evaluation.

Inversion of TUOV central map. Given parameters params $= (n, m, m_1, m_2, q)$, a *TUOV central map* in relation to params is $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ consisted of (n, m_1)-OV-polynomials $f_1, \ldots, f_{m_1}$, $(n, m, k - m_1)$-TOV-polynomial f_k for $k = m_1 + 1, \ldots, m_2$ and $(n, m - m_2 +$

$m_1 - 1)$-OV-polynomials $f_{m_2+1}, \ldots, f_m$, *i.e.*, $\mathcal{F} = (f_1, \ldots, f_m)^\top$. As for the triangular map, it lies in some (n, m)-OV-polynomial f and the TOV-polynomials f_k's for $k = m_1 + 1, \ldots, m_2$. More specifically, any (n, m)-OV-polynomial f can actually be rewritten as

$$f(\mathbf{x}) = x_{n-m+1} + h(\mathbf{x})$$

and an $(n, m, k - m_1)$-TOV-polynomial f_k can be rewritten as

$$f_k(\mathbf{x}) = x_{n-m+k-m_1+1} + g_k(x_{n-m+1}, \ldots, x_{n-m+k-m_1}) + h_k(\mathbf{x}),$$

where h and h_k's are (n, m)-OV-polynomials. So

$$\begin{aligned}&(f - h, f_{m_1+1} - h_{m_1+1}, \ldots, f_{m_2} - h_{m_2})\\ =&(x_{n-m+1}, x_{n-m+2} + g_{m_1+1}(x_{n-m+1}), \ldots,\\ &x_{n-m+m_2-m_1+1} + g_{m_2}(x_{n-m+1}, \ldots, x_{n-m+m_2-m_1}))\end{aligned}$$

which is obviously a triangular map in $x_{n-m+1}, \ldots, x_{n-m+m_2-m_1+1}$. Hence, to find a pre-image $\mathbf{x} \in \mathbb{F}_q^n$ of $\mathbf{y} \in \mathbb{F}_q^m$ under $\mathcal{F}$, there are three steps. First, we find $\mathbf{x}^{(1)}$ that vanishes $\mathbf{x}^{(212)}$ in h and $\mathbf{x}^{(2(k-m_1+1)2)}$ in h_k. Then we invert the triangular map to obtain $\mathbf{x}^{(2(m_2-m_1)1)}$. Lastly, we substitute values of $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2(m_2-m_1)1)}$, solving the system of linear equations in $\mathbf{x}^{(2(m_2-m_1)2)}$.

General description of TUOV scheme In general, a TUOV scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ can be constructed as follows:

- **params** $\leftarrow \mathsf{Setup}(1^\kappa)$.
  On security parameter, Setup algorithm generates system parameters params $= (n, m, m_1, m_2, q)$.
- $(\mathsf{pk} = \mathcal{P}, \mathsf{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T})) \leftarrow \mathsf{KeyGen}(\mathsf{params})$.
  The key generation algorithm outputs a random key pair $(\mathsf{pk} = \mathcal{P}, \mathsf{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T}))$. The secrete key $\mathsf{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T})$ consists of invertible affine transformations $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$, together with a TUOV central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ in relation to params. The public key $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ is a *TUOV map* in relation to params.
- $\boldsymbol{\sigma} \leftarrow \mathsf{Sign}(\mathsf{params}, \mathsf{sk}, \mu)$.
  The signing algorithm first computes the hash digest of the message $\mu \in \{0, 1\}^*$ as $\mathbf{z} \leftarrow \mathsf{Hash}(\mu)$, then successively finds preimages $\mathbf{y} \in \mathbb{F}_q^m$ of $\mathbf{z}$ under $\mathcal{S}$, $\mathbf{x} \in \mathbb{F}_q^n$ of $\mathbf{y}$ under the TUOV central map $\mathcal{F}$ as described above, and $\mathbf{w} \in \mathbb{F}_q^n$ of $\mathbf{x}$ under $\mathcal{T}$. Finally, it returns the signature $\sigma := \mathbf{w}$. Here $\mathsf{Hash} : \{0, 1\}^* \to \mathbb{F}_q^m$ denotes a hash function.
- $\boldsymbol{b} := \mathsf{Verify}(\mathsf{params}, \mathsf{pk}, \mu, \sigma)$
  The verification algorithm checks whether $\mathcal{P}(\sigma) = \mathsf{Hash}(\mu)$, and it returns accept if and only if the equality holds; otherwise, it returns reject.

For completeness, Figure 2 presents pseudocodes of the KeyGen, Sign and Verify algorithms. Notice that for each (T)OV-polynomial f_k in the TUOV central map $\mathcal{F} = (f_1, \ldots, f_m)^\top$, it has the unique representation as

$$f_k(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}_k \mathbf{x} + \mathbf{b}_k^\top \mathbf{x} + \gamma_k$$

where $\mathbf{A}_k \in \mathbb{F}_q^{n \times n}$ is upper-triangular, $\mathbf{b}_k \in \mathbb{F}_q^n$ and $\gamma_k \in \mathbb{F}_q$.

```
**KeyGen**($\mathsf{params} = (n, m, m_1, m_2, q)$):
Choose TUOV map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ in relation to $\mathsf{params}$ uniformly at random
Choose affine invertible maps $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$ uniformly at random
$\mathcal{P} := \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$
$\mathsf{pk} := \mathcal{P}$
$\mathsf{sk} := (\mathcal{S}, \mathcal{F}, \mathcal{T})$
**return** $(\mathsf{pk}, \mathsf{sk})$.

**Sign**($\mathsf{params}, \mathsf{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T}), \mu \in \{0,1\}^*$):
$\mathbf{z} \leftarrow \mathsf{Hash}(\mu)$                                   $\triangleright$ $\mathsf{Hash} : \{0,1\}^* \to \mathbb{F}_q^m$
$\mathbf{y} := \mathcal{S}^{-1}(\mathbf{z})$
**while** True **do**:
    $[\lambda_1, \ldots, \lambda_{m_1}]^\top \xleftarrow{\$} \mathbb{F}_q^{m_1}$
    $\widetilde{\mathbf{A}} := \sum_{k=1}^{m_1} \lambda_k \mathbf{A}_k, \widetilde{\mathbf{b}} := \sum_{k=1}^{m_1} \lambda_k \mathbf{b}_k, \begin{bmatrix} \widetilde{\gamma} & \widetilde{y} \end{bmatrix} := \sum_{k=1}^{m_1} \lambda_k \begin{bmatrix} \gamma_k & y_k \end{bmatrix}$
    **for** $l \in [m_2] \setminus [m_1]$ **do**:
        $\widetilde{\mathbf{A}}_l := \mathbf{A}_l, \widetilde{\mathbf{b}}_l := \mathbf{b}_l, \begin{bmatrix} \widetilde{\gamma}_l & \widetilde{y}_l \end{bmatrix} := \begin{bmatrix} \gamma_l & y_l \end{bmatrix}$
        **for** $k \in [m_1]$ **do**:
            $\lambda_{lk} \xleftarrow{\$} \mathbb{F}_q$
            $\widetilde{\mathbf{A}}_l \mathrel{+}= \lambda_{lk} \mathbf{A}_k, \widetilde{\mathbf{b}}_l \mathrel{+}= \lambda_{lk} \mathbf{b}_k, \begin{bmatrix} \widetilde{\gamma}_l & \widetilde{y}_l \end{bmatrix} \mathrel{+}= \lambda_{lk} \begin{bmatrix} \gamma_k & y_k \end{bmatrix}$
    Find a solution $\mathbf{x}^{(1)} = \mathbf{v} \in \mathbb{F}_q^{n-m}$ to the following system of equations:
        $\begin{cases} \begin{bmatrix} \widetilde{\mathbf{A}}^{(2)\top} \mathbf{x}^{(1)} + \widetilde{\mathbf{b}}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{0}_{m-1} \end{bmatrix} \\ \widetilde{\mathbf{A}}_k^{(2(l-m_1)2)\top} \mathbf{x}^{(1)} + \widetilde{\mathbf{b}}_k^{(2(l-m_1)2)} = \begin{bmatrix} 1 \\ \mathbf{0}_{m-k+m_1-1} \end{bmatrix} \text{ for } k \in [m_2] \setminus [m_1] \end{cases}$
    $\mathbf{u}^{(1)} := \begin{bmatrix} \widetilde{y} - \left( \mathbf{v}^\top \widetilde{\mathbf{A}}^{(1)} + \widetilde{\mathbf{b}}^{(1)\top} \right) \mathbf{v} - \widetilde{\gamma} \end{bmatrix}$
    **for** $k \in [m_2] \setminus [m_1]$ **do**:
        $u := \widetilde{y}_k - \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(d)} \end{bmatrix}^\top \begin{bmatrix} \widetilde{\mathbf{A}}_k^{(1)} & \widetilde{\mathbf{A}}_k^{(2d1)} \\ \mathbf{0} & \mathbf{A}_k^{(4d1)} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} - \begin{bmatrix} \widetilde{\mathbf{b}}_k^{(1)} \\ \widetilde{\mathbf{b}}_k^{(2d1)} \end{bmatrix}^\top \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} - \widetilde{\gamma}_k$
        $\mathbf{u}^{(1)} := \begin{bmatrix} \mathbf{u}^{(1)} \\ u \end{bmatrix}$
    $d := m_2 - m_1 + 1$
    Find a solution $\mathbf{x}^{(2d2)} = \mathbf{u}^{(2)} \in \mathbb{F}_q^{m-m_2+m_1-1}$ to the system of equations:
        $\begin{cases} \left( \mathbf{v}^\top \mathbf{A}^{(2d2)} + \mathbf{b}_k^{(2d2)\top} \right) \mathbf{x}^{(2d2)} = y_k - \gamma_k \\ - \left( \mathbf{v}^\top \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2d1)} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_k^{(1)} \\ \mathbf{b}_k^{(2d1)} \end{bmatrix}^\top \right) \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} \text{ for } k \in [m_1] \\ \left( \mathbf{v}^\top \mathbf{A}_k^{(2d2)} + \mathbf{b}_k^{(2d2)\top} \right) \mathbf{x}_k^{(2d2)} = y_k \\ - \left( \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix}^\top \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2d1)} \\ \mathbf{0} & \mathbf{A}_k^{(4d1)} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_k^{(1)} \\ \mathbf{b}_k^{(2d1)} \end{bmatrix}^\top \right) \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \end{bmatrix} \text{ for } k \in [m_2] \setminus [m_1] \end{cases}$
**return** $\mathbf{w} := \mathcal{T}^{-1} \left( \begin{bmatrix} \mathbf{v} \\ \mathbf{u}^{(1)} \\ \mathbf{u}^{(2)} \end{bmatrix} \right)$                   $\triangleright$ $\mathbf{w} \in \mathbb{F}_q^n$;

**Verify**($\mathsf{params}, \mathsf{pk} = \mathcal{P}, \mu, \sigma = \mathbf{w}$):
$\mathbf{z} \leftarrow \mathsf{Hash}(\mu)$
**return** $(\mathbf{z} == \mathcal{P}(\mathbf{w}))$.
```

Figure 2: The key generation, signing and verification algorithms of a general TUOV.

3.2 TUOV Scheme Description

In this section, we provide a detailed description to TUOV scheme with $m_2 = m_1$, including its key generation, signing and verification algorithms.

When $m_2 = m_1$, there is only one triangular variable, namely x_{n-m+1}, and the TUOV central map $\mathcal{F}$ only contains two layers of polynomials:

$$f_k(x_1, \cdots, x_n) = \sum_{i=1}^{n-m} \sum_{j=1}^{n} \alpha_{i,j}^{(k)} x_i x_j, \qquad k \in [m_1]$$

$$f_k(x_1, \cdots, x_n) = \sum_{i=1}^{n-m+1} \sum_{j=1}^{n} \alpha_{i,j}^{(k)} x_i x_j, \qquad k \in [m_1 + 1, m]$$

The first layer has m_1 polynomials, and the second has $m - m_1$ polynomials.

Conventions. In order to speed up key generation / signing process and compress the public key, we make the following restrictions on our TUOV instances:

- We can restrict ourselves to homogeneous maps $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$, since it is widely accepted that the homogeneous requirement does not weaken the security of multivariate public key cryptosystems (MPKC). In this manner the size of key could be slightly decreased.
- We can restrict that the linear maps $\mathcal{S}$ and $\mathcal{T}$ are of the form

$$\mathcal{S} = \begin{bmatrix} \mathbf{I}_{m_1} & \mathbf{S} \\ 0 & \mathbf{I}_{m-m_1} \end{bmatrix}$$
$$\mathcal{T} = \begin{bmatrix} \mathbf{I}_{n-m} & \mathbf{T}^{(1)} & \mathbf{T}^{(2)} \\ 0 & 1 & \mathbf{T}^{(3)} \\ 0 & 0 & \mathbf{I}_{m-1} \end{bmatrix} \tag{1}$$

where $\mathbf{S}$ and $\mathbf{T}^{(1)}$, $\mathbf{T}^{(2)}$ and $\mathbf{T}^{(3)}$ are random matrices. Since it does not shrink the key space much. For our special choice of $\mathcal{S}$ and $\mathcal{T}$ we have

$$\mathcal{S}^{-1} = \begin{bmatrix} \mathbf{I}_{m_1} & -\mathbf{S} \\ 0 & \mathbf{I}_{m-m_1} \end{bmatrix}$$
$$\mathcal{T}^{-1} = \begin{bmatrix} \mathbf{I}_{n-m} & -\mathbf{T}^{(1)} & \mathbf{T}^{(1)} \cdot \mathbf{T}^{(3)} - \mathbf{T}^{(2)} \\ 0 & 1 & -\mathbf{T}^{(3)} \\ 0 & 0 & \mathbf{I}_{m-1} \end{bmatrix}$$

For abbreviation, we set $\mathbf{T}^{(4)} := \mathbf{T}^{(1)} \cdot \mathbf{T}^{(3)} - \mathbf{T}^{(2)}$

We introduce an intermediate map $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$. Note that the three multivariate quadratic maps $\mathcal{F}$, $\mathcal{Q}$ and $\mathcal{P}$ can be represented by matrices in two different ways.

1. As a set of upper-triangular matrices $\{\mathbf{P}_i$ (resp. $\mathbf{F}_i$, $\mathbf{Q}_i$) $\}_{i \in [m]}$. We divide the $\mathbf{P}_i$ into submatrices as

$$\mathbf{P}_i = \begin{bmatrix} \mathbf{P}_i^{(1)} & \mathbf{P}_i^{(2)} & \mathbf{P}_i^{(3)} \\ \mathbf{0} & \mathbf{P}_i^{(5)} & \mathbf{P}_i^{(6)} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_i^{(9)} \end{bmatrix}, \tag{2}$$

where $\mathbf{P}_i^{(1)} \in \mathbb{F}_q^{(n-m)\times(n-m)}$, $\mathbf{P}_i^{(2)} \in \mathbb{F}_q^{(n-m)\times 1}$, $\mathbf{P}_i^{(3)} \in \mathbb{F}_q^{(n-m)\times(m-1)}$, $\mathbf{P}_i^{(5)} \in \mathbb{F}_q$, $\mathbf{P}_i^{(6)} \in \mathbb{F}_q^{1\times(m-1)}$, $\mathbf{P}_i^{(9)} \in \mathbb{F}_q^{(m-1)\times(m-1)}$. Similar notations carry over to $\mathbf{F}_i$, $\mathbf{Q}_i$.

2. As block Macaulay matrices $\overline{\mathbf{P}}$ (resp. $\overline{\mathbf{F}}$, $\overline{\mathbf{Q}}$) in $\mathbb{F}_q^{m\times D}$ where $D = n(n+1)/2$ denotes the number of nonzero elements in the upper-triangular matrix shown in (2). The i-th row of $\overline{\mathbf{P}}$ contains all the elements in $\mathbf{P}_i$ with first $\frac{(n-m)(n-m+1)}{2}$ elements contain all the elements in $\mathbf{P}_i^{(1)}$ and the second $n-m$ elements correspond to $\mathbf{P}_i^{(2)}$ and so on. Therefore,
$$\overline{\mathbf{P}} := \begin{bmatrix} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} & \overline{\mathbf{P}}_1^{(5)} & \overline{\mathbf{P}}_1^{(6)} & \overline{\mathbf{P}}_1^{(9)} \\ \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} & \overline{\mathbf{P}}_2^{(9)} \end{bmatrix}$$
where $\overline{\mathbf{P}}_1^{(k)}$ contains every matrix element in $\{\mathbf{P}_i^{(k)}\}_{i\in[m_1]}$ and $\overline{\mathbf{P}}_2^{(k)}$ contains every matrix element in $\{\mathbf{P}_i^{(k)}\}_{i\in[m_1+1,m]}$.

Both expressions are to be used in our algorithms.

Generation of $\mathcal{P}$. Inspired by the design of the CyclicRainbow scheme [10], the process of generating a public key from the secret key could be partially reversible in standard TUOV, which can be used to compress public key.

Since $\mathcal{Q} = \mathcal{F} \circ \mathcal{T}$, we get

$$\mathsf{UT}\left(\begin{bmatrix} \mathbf{I}_{n-m} & -\mathbf{T}^{(1)} & \mathbf{T}^{(4)} \\ \mathbf{0} & 1 & -\mathbf{T}^{(3)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{m-1} \end{bmatrix}^\top \begin{bmatrix} \mathbf{Q}_i^{(1)} & \mathbf{Q}_i^{(2)} & \mathbf{Q}_i^{(3)} \\ \mathbf{0} & \mathbf{Q}_i^{(5)} & \mathbf{Q}_i^{(6)} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_i^{(9)} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{n-m} & -\mathbf{T}^{(1)} & \mathbf{T}^{(4)} \\ \mathbf{0} & 1 & -\mathbf{T}^{(3)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{m-1} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{F}_i^{(1)} & \mathbf{F}_i^{(2)} & \mathbf{F}^{(3)} \\ \mathbf{0} & \mathbf{F}_i^{(5)} & \mathbf{F}_i^{(6)} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_i^{(9)} \end{bmatrix} \tag{3}$$

Hence,

$$\begin{aligned} \mathbf{F}_i^{(1)} &= \mathbf{Q}_i^{(1)} \\ \mathbf{F}_i^{(2)} &= -(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(1)} + \mathbf{Q}_i^{(2)} \\ \mathbf{F}_i^{(3)} &= (\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} - \mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} + \mathbf{Q}_i^{(3)} \\ \mathbf{F}_i^{(5)} &= \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(1)}\mathbf{T}^{(1)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)} + \mathbf{Q}_i^{(5)} \\ \mathbf{F}_i^{(6)} &= -\mathbf{T}^{(1)\top}(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(3)} + \mathbf{Q}_i^{(2)\top}\mathbf{T}^{(4)} + \mathbf{Q}_i^{(6)} \\ \mathbf{F}_i^{(9)} &= \mathsf{UT}\left(\mathbf{T}^{(4)\top}(\mathbf{Q}_i^{(1)}\mathbf{T}^{(4)} - \mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} + \mathbf{Q}_i^{(3)}) + \mathbf{T}^{(3)\top}(\mathbf{Q}_i^{(5)}\mathbf{T}^{(3)} - \mathbf{Q}_i^{(6)}) + \mathbf{Q}_i^{(9)}\right) \end{aligned} \tag{4}$$

By definition, $\mathbf{F}_i^{(9)} = \mathbf{0}$ for $i \in [m]$ and $\begin{bmatrix} \mathbf{F}_i^{(5)} & \mathbf{F}_i^{(6)} \end{bmatrix} = \mathbf{0}$ for $i \in [m_1]$. Then

$$\begin{aligned} \mathbf{Q}_i^{(5)} &= -\mathbf{T}^{(1)\top}\mathbf{Q}_i^{(1)}\mathbf{T}^{(1)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)} && i \in [m_1] \\ \mathbf{Q}_i^{(6)} &= \mathbf{T}^{(1)\top}(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(3)} - \mathbf{Q}_i^{(2)\top}\mathbf{T}^{(4)} && i \in [m_1] \\ \mathbf{Q}_i^{(9)} &= \mathsf{UT}\left(\mathbf{T}^{(4)\top}(-\mathbf{Q}_i^{(1)}\mathbf{T}^{(4)} + \mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} - \mathbf{Q}_i^{(3)}) + \mathbf{T}^{(3)\top}(-\mathbf{Q}_i^{(5)}\mathbf{T}^{(3)} + \mathbf{Q}_i^{(6)})\right) && i \in [m] \end{aligned} \tag{5}$$

It can be shown that $\mathcal{Q}$ is determined by $\mathcal{T}$, $\{\mathbf{Q}_i^{(1)}, \mathbf{Q}_i^{(2)}, \mathbf{Q}_i^{(3)}\}_{i\in[m]}$, $\{\mathbf{Q}_i^{(5)}, \mathbf{Q}_i^{(6)}\}_{i\in[m_1+1,m]}$ and so is $\mathcal{F}$. In other words, if we have $\mathcal{T}$, $\{\mathbf{Q}_i^{(1)}, \mathbf{Q}_i^{(2)}, \mathbf{Q}_i^{(3)}\}_{i\in[m]}$, $\{\mathbf{Q}_i^{(5)}, \mathbf{Q}_i^{(6)}\}_{i\in[m_1+1,m]}$, then we can compute $\{\mathbf{Q}_i^{(5)}, \mathbf{Q}_i^{(6)}\}_{i\in[m_1+1]}$, $\{\mathbf{Q}_i^{(9)}\}_{i\in[m]}$, $\{\mathbf{F}_i^{(1)}, \mathbf{F}_i^{(2)}, \mathbf{F}_i^{(3)}\}_{i\in[m]}$, $\{\mathbf{F}_i^{(5)}, \mathbf{F}_i^{(6)}\}_{i\in[m_1+1,m]}$ efficiently.

Since $\mathcal{P} = \mathcal{S} \circ \mathcal{Q}$, we have

$$\begin{aligned} &\begin{bmatrix} \overline{\mathbf{Q}}_1^{(1)} & \overline{\mathbf{Q}}_1^{(2)} & \overline{\mathbf{Q}}_1^{(3)} & \overline{\mathbf{Q}}_1^{(5)} & \overline{\mathbf{Q}}_1^{(6)} & \overline{\mathbf{Q}}_1^{(9)} \\ \overline{\mathbf{Q}}_2^{(1)} & \overline{\mathbf{Q}}_2^{(2)} & \overline{\mathbf{Q}}_2^{(3)} & \overline{\mathbf{Q}}_2^{(5)} & \overline{\mathbf{Q}}_2^{(6)} & \overline{\mathbf{Q}}_2^{(9)} \end{bmatrix} \\ = &\begin{bmatrix} \mathbf{I}_{m_1} & -\mathbf{S} \\ 0 & \mathbf{I}_{m-m_1} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} & \overline{\mathbf{P}}_1^{(5)} & \overline{\mathbf{P}}_1^{(6)} & \overline{\mathbf{P}}_1^{(9)} \\ \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} & \overline{\mathbf{P}}_2^{(9)} \end{bmatrix} \end{aligned} \tag{6}$$

It can be shown that $\{\overline{\mathbf{Q}}_j^{(1)}\ \overline{\mathbf{Q}}_j^{(2)}\overline{\mathbf{Q}}_j^{(3)}\}_{j=1,2}$, $\{\overline{\mathbf{Q}}_2^{(5)}, \overline{\mathbf{Q}}_2^{(6)}\}$ are determined by $\mathcal{S}$, $\{\overline{\mathbf{P}}_j^{(1)}\ \overline{\mathbf{P}}_j^{(2)}\overline{\mathbf{P}}_j^{(1)}\}_{j=1,2}$, $\{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\}$ Stated differently, if we have $\mathcal{S}, \mathcal{T}$, $\{\overline{\mathbf{P}}_j^{(1)}\ \overline{\mathbf{P}}_j^{(2)}\overline{\mathbf{P}}_j^{(1)}\}_{j=1,2}$, $\{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\}$, then we can first compute $\{\mathbf{Q}_i^{(1)}, \mathbf{Q}_i^{(2)}, \mathbf{Q}_i^{(3)}\}_{i\in[m]}$, $\{\mathbf{Q}_i^{(5)}, \mathbf{Q}_i^{(6)}\}_{i\in[m_1+1,m]}$, then $\{\mathbf{Q}_i^{(5)}, \mathbf{Q}_i^{(6)}\}_{i\in[m_1]}$, $\{\mathbf{Q}_i^{(9)}\}_{i\in[m]}$, $\{\mathbf{F}_i^{(1)}, \mathbf{F}_i^{(2)}, \mathbf{F}_i^{(3)}\}_{i\in[m]}$, $\{\mathbf{F}_i^{(5)}, \mathbf{F}_i^{(6)}\}_{i\in[m_1+1,m]}$, and finally, $\{\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}\}$, $\{\overline{\mathbf{P}}_j^{(9)}\}_{j=1,2}$. This procession is explained in Figure 3.

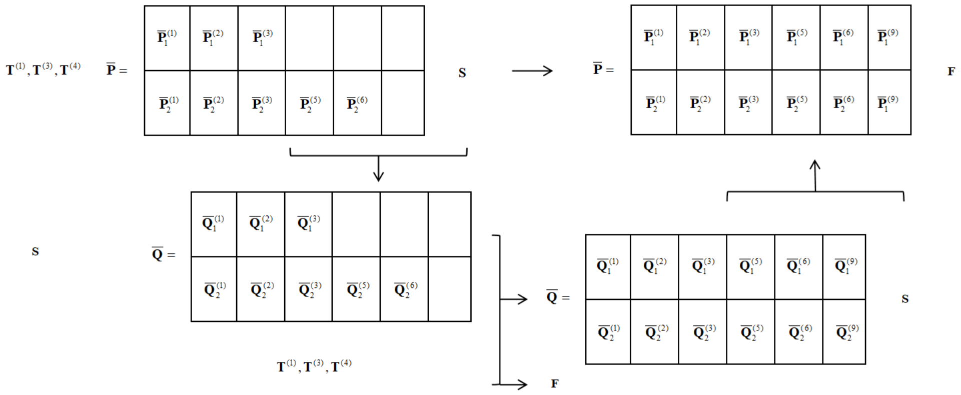
$\mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}$
$\overline{\mathbf{P}} =$
$\overline{\mathbf{P}}_1^{(1)}$ $\overline{\mathbf{P}}_1^{(2)}$ $\overline{\mathbf{P}}_1^{(3)}$
$\overline{\mathbf{P}}_2^{(1)}$ $\overline{\mathbf{P}}_2^{(2)}$ $\overline{\mathbf{P}}_2^{(3)}$ $\overline{\mathbf{P}}_2^{(5)}$ $\overline{\mathbf{P}}_2^{(6)}$
S
$\overline{\mathbf{P}} =$
$\overline{\mathbf{P}}_1^{(1)}$ $\overline{\mathbf{P}}_1^{(2)}$ $\overline{\mathbf{P}}_1^{(3)}$ $\overline{\mathbf{P}}_1^{(5)}$ $\overline{\mathbf{P}}_1^{(6)}$ $\overline{\mathbf{P}}_1^{(9)}$
$\overline{\mathbf{P}}_2^{(1)}$ $\overline{\mathbf{P}}_2^{(2)}$ $\overline{\mathbf{P}}_2^{(3)}$ $\overline{\mathbf{P}}_2^{(5)}$ $\overline{\mathbf{P}}_2^{(6)}$ $\overline{\mathbf{P}}_1^{(9)}$
F
S
$\overline{\mathbf{Q}} =$
$\overline{\mathbf{Q}}_1^{(1)}$ $\overline{\mathbf{Q}}_1^{(2)}$ $\overline{\mathbf{Q}}_1^{(3)}$
$\overline{\mathbf{Q}}_2^{(1)}$ $\overline{\mathbf{Q}}_2^{(2)}$ $\overline{\mathbf{Q}}_2^{(3)}$ $\overline{\mathbf{Q}}_2^{(5)}$ $\overline{\mathbf{Q}}_2^{(6)}$
$\mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}$
$\overline{\mathbf{Q}} =$
$\overline{\mathbf{Q}}_1^{(1)}$ $\overline{\mathbf{Q}}_1^{(2)}$ $\overline{\mathbf{Q}}_1^{(3)}$ $\overline{\mathbf{Q}}_1^{(5)}$ $\overline{\mathbf{Q}}_1^{(6)}$ $\overline{\mathbf{Q}}_1^{(9)}$
$\overline{\mathbf{Q}}_2^{(1)}$ $\overline{\mathbf{Q}}_2^{(2)}$ $\overline{\mathbf{Q}}_2^{(3)}$ $\overline{\mathbf{Q}}_2^{(5)}$ $\overline{\mathbf{Q}}_2^{(6)}$ $\overline{\mathbf{Q}}_2^{(9)}$
S
F

Figure 3: keygen

Inversion of $\mathcal{F}$. For any $\mathbf{t} = (t_1, \cdots, t_m) \in \mathbb{F}_q^m$, we want to find an $(s_1, \cdots, s_n)$ such that $\mathcal{F}(s_1, \cdots, s_n) = \mathbf{t}$. For the sake of convenience, we divide $(s_1, \cdots, s_n)$ into three part $\mathbf{v}, x, \mathbf{x}$, where $\mathbf{v} := (s_1, \cdots, s_{n-m})$, $x := s_{n-m+1}$, $\mathbf{x} := (s_{n-m+2}, \cdots, s_n)$. We want to compute them one by one. Firstly, careful analysis shows that if we can pick a random vector $\mathbf{v}$ satisfying the following conditions:

There exist $\{\lambda_i\}_{i\in[m_1]} \in \mathbb{F}_q^{m_1}$ such that $\sum_{i\in[m_1]} \lambda_i \mathbf{v}^\top \mathbf{F}_i^{(3)} = 0$, $\sum_{i\in[m_1]} \lambda_i \mathbf{v}^\top \mathbf{F}_i^{(2)} = 1$, Then we can recover x as follows:

$$x = \sum_{i\in[m_1]} \lambda_i (t_i - \mathbf{v}^\top \mathbf{F}_i^{(1)} \mathbf{v}),$$

Secondly, we can solve the following system of linear equations to get $\mathbf{x}$: Since the first m_1 equations have only $m_1 - 1$ linear independent equations, the following system is compatible with probability roughly $1 - 1/q$.

$$\begin{cases} \mathbf{v}^\top \mathbf{F}_i^{(3)} \mathbf{x} = t_i - \mathbf{v}^\top \mathbf{F}_i^{(1)} \mathbf{v} - \mathbf{v}^\top \mathbf{F}_i^{(2)} x & i \in [m_1] \\ (x\mathbf{F}_i^{(6)} + \mathbf{v}^\top \mathbf{F}_i^{(3)})\mathbf{x} = t_i - \mathbf{v}^\top \mathbf{F}_i^{(1)} \mathbf{v} - \mathbf{F}_i^{(5)} x^2 - \mathbf{v}^\top \mathbf{F}_i^{(2)} x & i \in [m_1 + 1, m] \end{cases}$$

If so, $(\mathbf{v}, x, \mathbf{x})$ is a desired pre-image obviously.

Finally, it remains to show how to recover a desired vector $\mathbf{v}$. In fact, we can find $\mathbf{v}$ by solving the system of linear equations

$$\sum_{i\in[m_1]} \lambda_i \begin{bmatrix} \mathbf{F}_i^{(2)} & \mathbf{F}_i^{(3)} \end{bmatrix}^\top \mathbf{v} = \begin{bmatrix} 1 \\ \mathbf{0}_{m-1} \end{bmatrix}.$$

Variants of TUOV. In order to achieve optimal efficiency under different circumstances, we introduce and implement three efficient variants of TUOV. We name them as `classic`, `pkc` and `pkc+skc`, which are in accordance with the three variants (`classic`, `circumzenithal`, `compressed`) of Rainbow signature scheme [11], respectively.
Before doing so, we give two key expansion procedures to achieve space/time efficiency tradeoff:

- ExpandPK: $\mathsf{cpk} \to \mathsf{upk}$. Here cpk (*resp.* upk) denotes compressed public key (*resp.* uncompressed public key).
- ExpandSK: $\mathsf{csk} \to \mathsf{usk}$. Similarly, csk (*resp.* usk) denotes compressed secret key (*resp.* uncompressed secret key).

In both cases, using the compressed key pair $(\mathsf{cpk}, \mathsf{csk})$ instead of uncompressed one $(\mathsf{upk}, \mathsf{usk})$ can lead to smaller memory requirement, at the cost of slower signing and verification speed.

Hence, the original key generation algorithm can be separated into three parts: KeyGen, ExpandPK, and ExpandSK where KeyGen only returns cpk and csk as output. This separation leads us to introduce and implement the following three TUOV variants.

- `classic`: the public/secret pair is $(\mathsf{upk}, \mathsf{usk})$. This means that both ExpandPK and ExpandSK are still parts of key generation algorithm. In this case, signing and verification procedure is quick but key sizes is very large.
- `pkc`: the public/secret pair is $(\mathsf{cpk}, \mathsf{usk})$. This means that ExpandSK is still part of key generation algorithm while ExpandPK is a part of verification algorithm. In this case, public key size is much smaller but verification procedure is slower in comparison with the `classic` variant.
- `pkc+skc`: the public/secret pair is $(\mathsf{cpk}, \mathsf{csk})$. This means that ExpandPK is a part of verification algorithm and ExpandSK is a part of signing algorithm. In this case, secret key size is extremely small but signing procedure is much slower in comparison with the `pkc` variant.

Description of `pkc+skc` variant. In the following part we will provide the detailed pseudocode of the `pkc+skc` variant to show how it scheme works. And the other two variants are straight forward. When dealing with `pkc+skc`, the secret key size is as short as $\mathsf{pk\_seed\_len} + \mathsf{sk\_seed\_len}$ bits and the current public parameter is

$$\mathsf{params} = (n, m, q, \mathsf{pk\_seed\_len}, \mathsf{sk\_seed\_len}).$$

Firstly, two seeds are picked uniformly at random, $\mathsf{seed_{sk}} \leftarrow \{0,1\}^{\mathsf{sk\_seed\_len}}$ and $\mathsf{seed_{pk}} \leftarrow \{0,1\}^{\mathsf{pk\_seed\_len}}$ in the key generation algorithm. Then $\mathsf{seed_{sk}}$ is used to generate $\{\mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}\}$, and $\mathsf{seed_{pk}}$ is used in the generation of $\{\overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{P}}_j^{(2)}, \overline{\mathbf{P}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\}$; thus $\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}$ as well as $\overline{\mathbf{P}}_2^{(9)}$ can be computed. And the key generation algorithm eventually returns $\mathsf{cpk} = (\mathsf{seed_{pk}}, \overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}, \overline{\mathbf{P}}_2^{(9)})$ and $\mathsf{csk} = (\mathsf{seed_{pk}}, \mathsf{seed_{sk}})$.

Then it comes to the signing algorithm. With ExpandSK, usk can be obtained with the outcomes of $\mathsf{Expand}_{\mathsf{sk}}(\mathsf{seed}_{\mathsf{sk}})$ and $\mathsf{Expand}_{\mathbf{P}}(\mathsf{seed}_{\mathsf{pk}})$. Here

$$\mathsf{usk} = (\mathsf{seed}_{\mathsf{sk}}, \mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}, \{\overline{\mathbf{F}}_j^{(1)} := \overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{F}}_j^{(2)}, \overline{\mathbf{F}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{F}}_2^{(5)}, \overline{\mathbf{F}}_2^{(6)}\}).$$

And with some delicate and intricate computation, an inversion of $\mathbf{t} = \mathsf{Hash}(\mu)$ under $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ will be eventually output as the signature σ.

In the verification algorithm, the full public key, *i.e.*, $\mathsf{upk} = \{\mathbf{P}_i\}_{i\in[m]}$ is obtained from the public key expansion ExpandPK(cpk) first. And it returns accept if and only if $\mathsf{Hash}(\mu) = [\mathbf{s}^\top \mathbf{P}_i \mathbf{s}]_{i\in[m]}$.

```
**KeyGen**(params):
$\mathsf{seed_{sk}} \leftarrow \{0,1\}^{\mathsf{sk\_seed\_len}}$
$\mathsf{seed_{pk}} \leftarrow \{0,1\}^{\mathsf{pk\_seed\_len}}$
$\{\mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}\} := \mathsf{Expand_{sk}}(\mathsf{seed_{sk}})$
$\{\overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{P}}_j^{(2)}, \overline{\mathbf{P}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\} := \mathsf{Expand}_{\mathbf{P}}(\mathsf{seed_{pk}})$
$\begin{bmatrix} \overline{\mathbf{Q}}_1^{(1)} & \overline{\mathbf{Q}}_1^{(2)} & \overline{\mathbf{Q}}_1^{(3)} \end{bmatrix} := \begin{bmatrix} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} \end{bmatrix} - \mathbf{S}\begin{bmatrix} \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} \end{bmatrix}$
$\begin{bmatrix} \overline{\mathbf{Q}}_2^{(1)} & \overline{\mathbf{Q}}_2^{(2)} & \overline{\mathbf{Q}}_2^{(3)} & \overline{\mathbf{Q}}_2^{(5)} & \overline{\mathbf{Q}}_2^{(6)} \end{bmatrix} := \begin{bmatrix} \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} \end{bmatrix}$
**for** $i = 1$ upto $m_1$ **do**
    $\mathbf{Q}_i^{(5)} := -\mathbf{T}^{(1)\top}\mathbf{Q}_i^{(1)}\mathbf{T}^{(1)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)}$
    $\mathbf{Q}_i^{(6)} := \mathbf{T}^{(1)\top}(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(3)} - \mathbf{Q}_i^{(2)\top}\mathbf{T}^{(4)}$
**for** $i = 1$ upto $m$ **do**
    $\mathbf{Q}_i^{(9)} := \mathsf{UT}\left(\mathbf{T}^{(4)\top}(-\mathbf{Q}_i^{(1)}\mathbf{T}^{(4)} + \mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} - \mathbf{Q}_i^{(3)}) + \mathbf{T}^{(3)\top}(-\mathbf{Q}_i^{(5)}\mathbf{T}^{(3)} + \mathbf{Q}_i^{(6)})\right)$
$\begin{bmatrix} \overline{\mathbf{P}}_1^{(5)} & \overline{\mathbf{P}}_1^{(6)} & \overline{\mathbf{P}}_1^{(9)} \end{bmatrix} := \begin{bmatrix} \overline{\mathbf{Q}}_1^{(5)} & \overline{\mathbf{Q}}_1^{(6)} & \overline{\mathbf{Q}}_1^{(9)} \end{bmatrix} + \mathbf{S}\begin{bmatrix} \overline{\mathbf{Q}}_2^{(5)} & \overline{\mathbf{Q}}_2^{(6)} & \overline{\mathbf{Q}}_2^{(9)} \end{bmatrix}$
$\overline{\mathbf{P}}_2^{(9)} := \overline{\mathbf{Q}}_2^{(9)}$
$\mathsf{cpk} := \left(\mathsf{seed_{pk}}, \{\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}, \overline{\mathbf{P}}_2^{(9)}\}\right)$
$\mathsf{csk} := (\mathsf{seed_{pk}}, \mathsf{seed_{sk}})$
**return** $(\mathsf{cpk}, \mathsf{csk})$.

**ExpandSK**(params, csk):                    ▷ $\mathsf{csk} = (\mathsf{seed_{sk}}, \mathsf{seed_{pk}})$
$\{\mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}\} := \mathsf{Expand_{sk}}(\mathsf{seed_{sk}})$
$\{\overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{P}}_j^{(2)}, \overline{\mathbf{P}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\} := \mathsf{Expand}_{\mathbf{P}}(\mathsf{seed_{pk}})$
$\begin{bmatrix} \overline{\mathbf{Q}}_1^{(1)} & \overline{\mathbf{Q}}_1^{(2)} & \overline{\mathbf{Q}}_1^{(3)} \end{bmatrix} := \begin{bmatrix} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} \end{bmatrix} - \mathbf{S}\begin{bmatrix} \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} \end{bmatrix}$
$\begin{bmatrix} \overline{\mathbf{Q}}_2^{(1)} & \overline{\mathbf{Q}}_2^{(2)} & \overline{\mathbf{Q}}_2^{(3)} & \overline{\mathbf{Q}}_2^{(5)} & \overline{\mathbf{Q}}_2^{(6)} \end{bmatrix} := \begin{bmatrix} \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} \end{bmatrix}$
**for** $i = 1$ upto $m$ **do**
    $\mathbf{F}_i^{(2)} := -(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(1)} + \mathbf{Q}_i^{(2)}$
    $\mathbf{F}_i^{(3)} := (\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} - \mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} + \mathbf{Q}_i^{(3)}$
**for** $i = m_1 + 1$ upto $m$ **do**
    $\mathbf{F}_i^{(5)} := \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(1)}\mathbf{T}^{(1)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)} + \mathbf{Q}_i^{(5)}$
    $\mathbf{F}_i^{(6)} := -\mathbf{T}^{(1)\top}(\mathbf{Q}_i^{(1)} + \mathbf{Q}_i^{(1)\top})\mathbf{T}^{(4)} + \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(2)}\mathbf{T}^{(3)} - \mathbf{T}^{(1)\top}\mathbf{Q}_i^{(3)} + \mathbf{Q}_i^{(2)\top}\mathbf{T}^{(4)} + \mathbf{Q}_i^{(6)}$
$\mathsf{usk} := \left(\mathsf{seed_{sk}}, \mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}, \{\overline{\mathbf{F}}_j^{(1)} := \overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{F}}_j^{(2)}, \overline{\mathbf{F}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{F}}_2^{(5)}, \overline{\mathbf{F}}_2^{(6)}\}\right)$
**return** usk.

**ExpandPK**(params, cpk):                    ▷ $\mathsf{cpk} = \left(\mathsf{seed_{pk}}, \{\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}, \overline{\mathbf{P}}_2^{(9)}\}\right)$
$\{\overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{P}}_j^{(2)}, \overline{\mathbf{P}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\} := \mathsf{Expand}_{\mathbf{P}}(\mathsf{seed_{pk}})$
$\mathsf{upk} := \begin{bmatrix} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} & \overline{\mathbf{P}}_1^{(5)} & \overline{\mathbf{P}}_1^{(6)} & \overline{\mathbf{P}}_1^{(9)} \\ \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} & \overline{\mathbf{P}}_2^{(9)} \end{bmatrix}$
**return** upk.
```

Figure 4: The key generation algorithms of `pkc+skc` variant of TUOV.

```
$\mathbf{Sign}(\mathsf{params}, \mathsf{csk}, \mu)$:
usk := $\mathbf{ExpandSK}(\mathsf{csk})$    ▷ $\mathsf{sk} := \big(\mathsf{seed}_{\mathsf{sk}}, \mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}, \{\mathbf{F}_i\}_{i\in[m]}\big)$
$\mathbf{t} := \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix} \leftarrow \mathsf{Hash}(\mu)$    ▷ $\mathbf{t}_1 \in \mathbb{F}_q^{m_1}, \mathbf{t}_2 \in \mathbb{F}_q^{m-m_1}$.
$\tilde{\mathbf{t}} := \begin{bmatrix} \mathbf{t}_1 + \mathbf{S}\mathbf{t}_2 \\ \mathbf{t}_2 \end{bmatrix}$
for ctr = 0 upto 255 do
    $\boldsymbol{\lambda} := (\lambda_1, \cdots, \lambda_{m_1}) := \mathsf{Expand}_{\boldsymbol{\lambda}}(\mu \| \mathsf{seed}_{\mathsf{sk}} \| \mathsf{ctr})$    ▷ $\boldsymbol{\lambda} \in \mathbb{F}_q^{m_1}$
    $\mathbf{M} := \lambda_1 \begin{bmatrix} \mathbf{F}_1^{(2)} & \mathbf{F}_1^{(3)} \end{bmatrix} + \cdots + \lambda_{m_1} \begin{bmatrix} \mathbf{F}_{m_1}^{(2)} & \mathbf{F}_{m_1}^{(3)} \end{bmatrix}$    ▷ $\mathbf{M} = \boldsymbol{\lambda} \begin{bmatrix} \overline{\mathbf{F}}_1^{(2)} & \overline{\mathbf{F}}_1^{(3)} \end{bmatrix}$
    if $\mathbf{M}^\top \mathbf{v} = \begin{bmatrix} 1 \\ \mathbf{0}_{m-1} \end{bmatrix}$ is solvable then
        $\mathbf{v} \leftarrow$ solution space of $\mathbf{M}^\top \mathbf{v} = \begin{bmatrix} 1 \\ \mathbf{0}_{m-1} \end{bmatrix}$    ▷ $\mathbf{v}' \in \mathbb{F}_q^{n-m}$
        $x := \boldsymbol{\lambda} \left[ \tilde{t}_i - \mathbf{v}^\top \mathbf{F}_i^{(1)} \mathbf{v} \right]_{i\in[m_1]}$
        Choose $k \in [m_1]$ such that $\lambda_k \neq 0$
        $\mathbf{L} := \mathbf{0}_{(m-1)\times(m-1)}$
        $\mathbf{y} := \mathbf{0}_{m-1}$
        for $i = 1$ upto $k - 1$ do
            Set $i$-th row of $\mathbf{L}$ to $\mathbf{v}^\top \mathbf{F}_i^{(3)}$
            Set $i$-th row of $\mathbf{y}$ to $\mathbf{v}^\top \mathbf{F}_i^{(1)} \mathbf{v} + \mathbf{v}^\top \mathbf{F}_i^{(2)} x$
        for $i = k$ upto $m_1 - 1$ do
            Set $i$-th row of $\mathbf{L}$ to $\mathbf{v}^\top \mathbf{F}_{i+1}^{(3)}$
            Set $i$-th row of $\mathbf{y}$ to $\mathbf{v}^\top \mathbf{F}_{i+1}^{(1)} \mathbf{v} + \mathbf{v}^\top \mathbf{F}_{i+1}^{(2)} x$
        for $i = m_1 - 1$ upto $m - 1$ do
            Set $i$-th row of $\mathbf{L}$ to $\mathbf{v}^\top \mathbf{F}_{i+1}^{(3)} + x\mathbf{F}_{i+1}^{(6)}$
            Set $i$-th row of $\mathbf{y}$ to $\mathbf{v}^\top \mathbf{F}_{i+1}^{(1)} \mathbf{v} + \mathbf{v}^\top \mathbf{F}_{i+1}^{(2)} x + \mathbf{F}_{i+1}^{(5)} x^2$
        if $\mathbf{L}$ is invertible then
            remove the $k$-th row of $\tilde{\mathbf{t}}$
            Solve $\mathbf{L}\mathbf{x} = \tilde{\mathbf{t}} - \mathbf{y}$ for $\mathbf{x}$    ▷ $\mathbf{x} \in \mathbb{F}_q^{m-1}$
            $\mathbf{s} := \begin{bmatrix} \mathbf{v} \\ 0 \\ \mathbf{0}_{m-1} \end{bmatrix} + \begin{bmatrix} \mathbf{T}^{(1)} \\ 1 \\ \mathbf{0}_{m-1} \end{bmatrix} x + \begin{bmatrix} \mathbf{T}^{(4)} \\ \mathbf{T}^{(3)} \\ \mathbf{I}_{m-1} \end{bmatrix} \cdot \mathbf{x}$    ▷ $\mathbf{s} \in \mathbb{F}_q^n$
            $\sigma := \mathbf{s}$
            return $\sigma$
return $\perp$.

$\mathbf{Verify}(\mathsf{params}, \mathsf{cpk}, (\mu, \sigma = \mathbf{s}))$:
upk := $\mathbf{ExpandPK}(\mathsf{cpk})$    ▷ $\mathsf{upk} = \{\mathbf{P}_i\}_{i\in[m]}$
$\mathbf{t} \leftarrow \mathsf{Hash}(\mu)$
return $\Big(\mathbf{t} == \left[\mathbf{s}^\top \mathbf{P}_i \mathbf{s}\right]_{i\in[m]}\Big)$.
```

Figure 5: The signing and verification algorithms of `pkc+skc` variant of TUOV.

3.3 Recommended Parameter Sets

To accommodate different security needs, we propose four sets of recommended parameters for TUOV and implement them in our submission, as shown in Table 2. Each set of parameters is applicable to all TUOV variants proposed in Section 3.2, with pk_seed_len always being 128 and sk_seed_len being 256. The variation in these sets is based on the choice of (n, m, m_1, q) :

- For NIST security level 1, we suggest two sets of parameters: `tuov-Ip`, which employs $\mathbb{F}_{256}$ and has smaller key sizes, and `tuov-Is`, which operates over $\mathbb{F}_{16}$ and has shorter signatures.
- For NIST security level 3, we propose `tuov-III` as the recommended parameter set.
- Finally, we recommend `tuov-V` for NIST security level 5.

In Section 4, we conduct a concrete security analysis of the computational hardness of TUOV and provide evidence that each set of recommended parameters can achieve its intended security level, respectively.

We choose $\mathbb{F}_{256}$ and $\mathbb{F}_{16}$ as our finite fields, with one element in $\mathbb{F}_{256}$ occupying one byte and two elements in $\mathbb{F}_{16}$ occupying one byte. Note that if the $\mathbb{F}_{16}$ vector has an odd length, we pad the last element with a zero. For specific calculation details of key/signature lengths, please refer to the Section 5.

Table 2: Recommended parameter sets and and the corresponding key/signature sizes for TUOV. Note that in each parameter set, we have pk_seed_len = 128, and sk_seed_len = 256.

	NIST Security Level	(n, m, m_1, q)	$\lvert\mathsf{upk}\rvert$ (bytes)	$\lvert\mathsf{usk}\rvert$ (bytes)	$\lvert\mathsf{cpk}\rvert$ (bytes)	$\lvert\mathsf{csk}\rvert$ (bytes)	$\lvert\sigma\rvert$ (bytes)
`tuov-Ip`	1	(112, 44, 22, 256)	278 432	239 391	42 608	48	112
`tuov-Is`	1	(160, 64, 32, 16)	412 160	350 272	65 552	48	80
`tuov-III`	3	(184, 72, 36, 256)	1 225 440	1 048 279	186 640	48	184
`tuov-V`	5	(244, 96, 48, 256)	2 869 440	2 443 711	442 384	48	244

3.4 Choice of Symmetric Primitives

We use various hash functions and pseudo-random functions in the implementation of TUOV. Concretely, we make use of shake256 [12] to process both public and secret data in the performance non-critical functions, namely Hash, $\mathsf{Expand}_{\mathbf{v}}$ and $\mathsf{Expand}_{\mathsf{sk}}$. On the other hand, we instantiate the performance critical function $\mathsf{Expand}_{\mathsf{pk}}$ using aes128 [13] with counter [14], *i.e.*, aes128ctr.

By taking advantage of the properties of aes128ctr, there is some technique to achieve faster implementations. We will show the implementation details of these function in the following context.

- **Hash(μ) :** $\{0,1\}^* \leftarrow \mathbb{F}_q^m$
  This function hashes a message μ to the target vector $\mathbf{t}$. The size of the target vector is $m \cdot \lceil \log_2 q \rceil$ bits.
- **$\mathsf{Expand}_{\lambda}(\mu \| \mathsf{seed}_{\mathsf{sk}} \| \mathsf{ctr})$:** $\{0,1\}^* \times \{0,1\}^{\mathsf{sk\_seed\_len}} \to \mathbb{F}_q^{n-2m+m_1}$
  This function samples a partial vinegar vector (last $n - 2m$ entries of the $\mathbf{v}$ in $\mathbb{F}_q^{n-2m}$) and a vector $\lambda = (\lambda_1, \ldots, \lambda_{m_1}) \in \mathbb{F}_q^{m_1}$. And the size of its output in total is $(n - 2m + m_1) \cdot \lceil \log_2 q \rceil$ bits.

- **Expand$_{\mathsf{sk}}$(seed$_{\mathsf{sk}}$) :** $\{0,1\}^{\mathsf{sk\_seed\_len}} \to \mathbb{F}_q^{m_1(m-m_1)+(n-m)(m-1)+n-1}$
  This function expands the 32-byte secret key seed to a matrix $\mathbf{S}$, and three matrices $\{\mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}\}$ sequentially. Here $\mathbf{S}$ relates to $\mathcal{S}^{-1} = \mathcal{S}$ while $\{\mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}\}$ relate to $\mathcal{T}^{-1}$. And the size of its output in total is $[m_1(m-m_1) + (n-m)(m-1) + n - 1] \cdot \lceil \log_2 q \rceil$ bits.

- **Expand$_{\mathbf{P}}$**(seed$_{\mathsf{pk}}$) : $\{0,1\}^{\mathsf{pk\_seed\_len}} \to \mathbb{F}_q^{m\cdot[v(v+1)/2+vm]+(m-m_1)m}$
  This function expands the 16-byte public key seed to eight column-major Macaulay matrices $\{\overline{\mathbf{P}}_j^{(1)}, \overline{\mathbf{P}}_j^{(2)}, \overline{\mathbf{P}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{P}}_2^{(5)}, \overline{\mathbf{P}}_2^{(6)}\}$. And the size of its output in total is $\{m \cdot [v(v+1)/2 + vm] + (m-m_1)m\} \cdot \lceil \log_2 q \rceil$ bits.

- When it comes to implementing the four symmetric primitives (Hash, Expand$_{\mathbf{v}}$, Expand$_{\mathsf{sk}}$, Expand$_{\mathbf{P}}$) efficiently, we resort to the OpenSSL library to gain access to the standard cryptographic primitives, to be more precise, shake256 and aes128 in our cases. In addition, we may make use of the round-reduced AES, *i.e.*, 4-round aes128ctr [Gue10], for faster implementation of Expand$_{\mathbf{P}}$.

4 Security Analysis

This section conducts the security analysis of TUOV. To be precise, we perform a comprehensive analysis from two perspectives:

- First, we analyze the EUF-CMA of the TUOV. While we fail to present a reduction between the TUOV problem and the EUF-CMA security of the TUOV scheme, we can base the hardness of the intermediate problem, *i.e.*the TUOV problem, on that of the MQ problem,
- Second, we perform a comprehensive security strength analysis that takes into account the most important attacks against TUOV. This includes, but is not limited to, the analysis of various forms of direct, Kipnis-Shamir, intersection, and quantum attacks, as well as the potential strategies to counter such threats. Our goal is to ensure that our recommended parameter set is robust against these threats and thus meets the respective security strength level required by NIST.

4.1 Security Reductions

4.1.1 Security Problems

Definition 5 (MQ map)**.** An (n, q)*-MQ-polynomial* f over $\mathbb{F}_q$ is defined as

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{i,j}x_ix_j + \sum_{i=1}^{n}\beta_i x_i + \gamma$$

and has unique representation as $\mathbf{x}^\top\mathbf{A}\mathbf{x}+\mathbf{b}^\top\mathbf{x}+\gamma$, where $\mathbf{A} \in \mathbb{F}_q^{n\times n}$ is anupper-triangular matrix, $\mathbf{b} \in \mathbb{F}_q^n$ is a vector. An (n, m, q)*-MQ-map* $\mathcal{M} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ consists with (n, q)-MQ polynomials $f^{(1)}, \dots, f^{(m)}$, i.e. $\mathcal{M} = (f_1, \dots, f_m)^\top$.

Definition 6 (MQ problem hardness)**.** Fix $\mathsf{Setup}(\cdot)$ that outputs (n, m, q) on input 1^κ. *MQ problem* in relation to $\mathsf{Setup}(\cdot)$ is (t, ε)-hard if there exists no algorithm that, given security parameter κ, $\mathsf{params} = (n, m, q) \leftarrow \mathsf{Setup}(1^\kappa)$ and a random params-MQ-map $\mathcal{M} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ in relation to params, on input $\mathbf{y} := \mathcal{M}(\mathbf{w})$ with $\mathbf{w} \overset{\$}{\leftarrow} \mathbb{F}_q^n$, outputs $\mathbf{w}'$ such that $\mathcal{M}(\mathbf{w}') = \mathbf{y}$ with probability no less than $\varepsilon(\kappa)$ in processing time $t(\kappa)$.

Definition 7 (TUOV problem hardness)**.** Fix $\mathsf{Setup}(\cdot)$ that outputs $\mathsf{params} = (n, m, m_1, m_2, q)$ on input 1^κ. TUOV problem in relation to $\mathsf{Setup}(\cdot)$ is (t, ε)-hard if there exists no algorithm that, given params and a random TUOV map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ in relation to which, on input $\mathbf{z} = \mathcal{P}(\mathbf{w})$ with $\mathbf{w} \overset{\$}{\leftarrow} \mathbb{F}_q^n$, outputs $\mathbf{w}'$ such that $\mathcal{P}(\mathbf{w}') = \mathbf{z}$ with probability no less than $\varepsilon(\kappa)$ in processing time $t(\kappa)$.

Definition 8 (TUOV scheme security)**.** A TUOV scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is $(t, \varepsilon, q_H, q_s)$-secure under random oracle model if there exists no algorithm that on security parameter κ, $\mathsf{params} = (n, m, m_1, m_2, q) \leftarrow \mathsf{Setup}(1^\kappa)$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, with at most q_H hash queries and q_s signing queries, outputs $(\mu, \mathbf{w})$ such that $\mathsf{Verify}(\mathsf{pk}, \mu, \mathbf{w}) = \mathsf{accept}$ in processing time $t(\kappa)$ with probability no less than $\varepsilon(\kappa)$.

4.1.2 Reduce MQ problem to TUOV problem

Theorem 1. *Given* $\mathsf{Setup}(\cdot)$ *that outputs* $\mathsf{params} = (n = \frac{1}{2} \cdot m^2, m, m_1 = \frac{1}{2} \cdot m, m_2 = \frac{3}{4} \cdot m, q)$ *on input* 1^κ *and its restriction* Setup' *that outputs* $\mathsf{params}' = (n = \frac{1}{2} \cdot m^2, m, q)$*. If MQ map in relation to* $\mathsf{Setup}'(\cdot)$ *is* (t, ε)*-hard, then TUOV map in relation to* $\mathsf{Setup}(\cdot)$ *is* (t, ε)*-hard.*

Proof. Without loss of generality, we consider quadrtic polynomials only with their quadratic part. Roughly speaking, for arbitrary MQ map $\mathcal{M} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, if there exists an invertible affine transformation $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ and a TUOV central map $\mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ such that $\mathcal{M} = \mathcal{F} \circ \mathcal{T}$, then the MQ input $(\mathcal{M}, \mathbf{y})$ can be modified into TUOV input $(\mathcal{P} = \mathcal{S} \circ \mathcal{M}, \mathbf{z} = \mathcal{S}(\mathbf{y}))$ for arbitrary invertible affine transformation $\mathcal{S} : \mathbb{F}_q^m \to \mathbb{F}_q^m$.

Hence, if there exists an algorithm $\mathcal{A}$ that can break TUOV problem in relation to $\mathsf{Setup}(\cdot)$, then we can construct an algorithm $\mathcal{B}$ that on MQ input $(\mathcal{M}, \mathbf{y})$, can randomly sample an $\mathcal{S}$ and makes query $(\mathcal{P} = \mathcal{S} \circ \mathcal{M}, \mathbf{z} = \mathcal{S}(\mathbf{y}))$ on $\mathcal{S}$ and outputs the $\mathbf{w}'$ returns by $\mathcal{A}$. Hence the theorem is proved.

Since $\mathcal{T}$ is invertible affine transformation, it suffices to find $\mathcal{Q} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ such that $\mathcal{M} \circ \mathcal{Q}$ is TUOV central map. Then we can claim that $\mathcal{T} = \mathcal{Q}^{-1}$, so $\mathcal{F} := \mathcal{M} \circ \mathcal{Q}$ is an equivalent TUOV central map that we are looking for. Notice our goal is to prove the existence, with abuse of symbols, consider $\mathbf{Q} = \begin{bmatrix} \mathbf{Q}^{(1)} & \mathbf{Q}^{(2)} \\ \mathbf{0}_{m \times (n-m)} & \mathbf{I}_m \end{bmatrix} \in \mathbb{F}_q^{n \times n}$ as the associate upper triangular matrix of $\mathcal{Q}$ in orthonormal basis, and let $\mathbf{M}_k = \begin{bmatrix} \mathbf{M}_k^{(1)} & \mathbf{M}_2^{(2)} \\ \mathbf{0} & \mathbf{M}_k^{(4)} \end{bmatrix}$ be the coefficient matrix of $\mathcal{M}$'s k-th polynomial. Notice that

$$\mathbf{Q}^\top \mathbf{M}_k \mathbf{Q} = \begin{bmatrix} \mathbf{Q}^{(1)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(1)} & \mathbf{Q}^{(1)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(2)} + \mathbf{Q}^{(1)\top} \mathbf{M}_k^{(2)} \\ \mathbf{Q}^{(2)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(1)} & \mathbf{Q}^{(2)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(1)} + \mathbf{Q}^{(2)\top} \mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)} \end{bmatrix}$$

and we want it to be somewhat equal to some upper triangular $\mathbf{A}_k$ with

$$\mathbf{A}_k = \begin{cases} \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2)} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, & k \in [m_1] \\ \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2d1)} & \mathbf{A}_k^{(2d2)} \\ \mathbf{0} & \mathbf{A}_k^{(4d1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, & d = k - m_1 \text{ and } k \in [m_2] \setminus [m_1] \\ \begin{bmatrix} \mathbf{A}_k^{(1)} & \mathbf{A}_k^{(2(m_2-m_1+1)1)} & \mathbf{A}_k^{(2(m_2-m_1+1)2)} \\ \mathbf{0} & \mathbf{A}_k^{(4(m_2-m_1+1)1)} & \mathbf{A}_k^{(4(m_2-m_1+1)2)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, & k \in [m] \setminus [m_2] \end{cases}$$

then our undetermined variables are terms in $\mathbf{Q}^{(2)} \in \mathbb{F}_q^{(n-m) \times m}$. So there are $(n-m) \times m \approx \frac{1}{2} m^3$ variables.

Now we have to count number of equations:

- for $k \in [m_1]$, $\mathsf{UT}\left(\mathbf{Q}^{(2)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(1)} + \mathbf{Q}^{(2)\top} \mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)}\right) = \mathbf{0}_{m \times m}$ brings $\frac{(m+1)m}{2}$ equations. So total number of equations here is $\frac{m_1(m+1)m}{2}$.
- for $k \in [m_2] \setminus [m_1]$, let $d = k - m_1$

$$\mathsf{UT}\left(\mathbf{Q}^{(2)\top} \mathbf{M}_k^{(1)} \mathbf{Q}^{(1)} + \mathbf{Q}^{(2)\top} \mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)}\right) = \begin{bmatrix} \mathbf{A}_k^{(4d1)} & \mathbf{0}_{d \times (m-d)} \\ \mathbf{0}_{(m-d) \times d} & \mathbf{0}_{m-d} \end{bmatrix}$$

brings $\frac{(m+1)m}{2} - \frac{d(d-1)}{2}$ equations. So total number of equations here is

$$\sum_{d=1}^{m_2-m_1} \left(\tfrac{(m+1)m}{2} - \tfrac{d(d-1)}{2}\right) = \tfrac{1}{6}(m_2 - m_1)\left(3m(m+1) - (m_2 - m_1)^2 + 1\right).$$

- for $k \in [m] \setminus [m_2]$,

$$\mathsf{UT}\left(\mathbf{Q}^{(2)\top}\mathbf{M}_k^{(1)}\mathbf{Q}^{(1)} + \mathbf{Q}^{(2)\top}\mathbf{M}_k^{(2)} + \mathbf{M}_k^{(4)}\right) = \begin{bmatrix} \mathbf{A}_k^{(4)} & \mathbf{A}_k^{(23)} \\ \mathbf{0} & \mathbf{0}_{m-m_2+m_1-1} \end{bmatrix}$$

  brings $\frac{1}{2}(m - m_2 + m_1 - 1)(m - m_2 + m_1)$ equations. So total number of equations here is $\frac{(m-m_2)}{2}(m - m_2 + m_1 - 1)(m - m_2 + m_1)$.

So the total number of equations to solve is

$$\begin{aligned} &\tfrac{m_1}{2}(m+1)m + \tfrac{m_2-m_1}{6}\left(3m(m+1) - (m_2-m_1)^2 + 1\right) \\ &+\tfrac{m-m_2}{2}(m - m_2 + m_1 - 1)(m - m_2 + m_1). \end{aligned} \tag{7}$$

Let $m_1 = (1-\beta)m$ and $m_2 - m_1 = \beta\gamma m$, then $m - m_2 + m_1 = (1-\beta\gamma)m$. And coefficient of m^3 in Equation (7) is

$$\tfrac{1}{2} + \beta^2\gamma\left(\tfrac{2}{3}\beta\gamma^2 - 1 + \tfrac{1}{2}\beta + \gamma\right). \tag{8}$$

If we evaluate $\beta = \gamma = \frac{1}{2}$, then we have roughly $\frac{11}{24}m^3$ equations. With underdetermined assumption, as long as $n \geq \frac{11}{24} \cdot m^2$, there exists invertible affine transformation $\mathcal{Q}$ such that $\mathcal{M} \circ \mathcal{Q}$ is TUOV central map. □

4.1.3 Reduce TUOV problem to TUOV scheme

Claim 1. *Given* $\mathsf{Setup}(\cdot)$ *that outputs* (n, m, m_1, m_2, q) *on input* 1^κ*. If TUOV problem in relation to* $\mathsf{Setup}(\cdot)$ *is* (t, ε)*-hard, then TUOV scheme* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* $(t, \varepsilon, q_H, q_s)$*-secure under random oracle model.*

The TUOV scheme, like its UOV predecessor, is constructed according to the hash-and-sign paradigm. As such, its security proof involves a standard approach: manipulating the hash value to avoid the computationally expensive inversion process, thus redirecting the procedure to more manageable operations, which consequently reduces a hard problem to the security of the scheme. However, as part of such proofs, it is crucial to show that a set of variables are indistinguishable in their distribution between the real signing process and a simulated one. Specifically, it is necessary to prove that the joint distribution of the public key, hash values, and signature values, *i.e.*$(\mathsf{pk}, \{h_i\}, \{\sigma_j\})$ where h_i's denote Hash query responses and σ_j's denote signing query responses, is indistinguishable under the two scenarios.

Computationally, it turns out that in the real scenario, the possible values of the vinegar variables occupy a tiny fraction of the total vinegar space $(q^{(-2m+m_2+m_1)(m_2-m_1+1)/2})$, making the above distributions statistically distinguishable. The typical fallback is to study computational indistinguishability, but this often introduces an additional assumption. A review of the history of MPKCs has underscored the need for extreme caution when introducing hard problems, as doing otherwise can lead to a false sense of security. As a matter of fact, we have proven that the real values of the vinegar variables are indistinguishable from uniform sampling in the vinegar space, which we believe provides an initial level of confidence in the security of TUOV. Nevertheless, we welcome any possible adjustments to the scheme that could lead to a stronger proof of security.

4.2 Attack Scenarios

In this section we introduce the state-of-the-art attacks against TUOV scheme, and analyze the hard estimation result of the four sets of recommended parameters of TUOV.

Similar to most of the cryptosystems in MPKC, we have not presented a formal security proof which reduces certain well-known “hard” mathematical problem(s) say, the MQ problem, to the security of TUOV. Here, in this documentation the security analysis for TUOV is carried out by listing some of the critical attacks against TUOV that may influence its concrete hardness estimation result. Our confidence in the security of TUOV lies in the facts that UOV remains secure after more than twenty years of cryptanalysis, and TUOV seems to be more secure than UOV, and that there is a solid theoretical foundation on the concrete hardness estimation of practical attacks against MPKC such that the theoretical hardness estimation of TUOV matches the experimental results consistently.

Historically, those attacks against TUOV are usually classified into two types: The *key-recovery attacks* aims to recover the secret key from the given public key, *e.g.*, the Kipnis-Shamir attack [8], the Intersection attack [15] and the MinRank attack; and the *forgery attacks* that aims to forge a message/signature pair passing the verification test, *e.g.*, the direct attack. The collision attack is not considered in this submission, because when the hash function is fixed, every message has a unique hash digest, making the collision attack against TUOV far from practical.

4.2.1 Direct Attack

The most straightforward attack against TUOV, (and even against most of the MPKC cryptosystems) is the direct attack, where the attacker aims to solve an instance of the MQ problem associated with the public key $\mathcal{P}$. In the direct attack, the attacker first chooses a message $\mu^* \in \{0,1\}^*$ and a salt $\mathsf{salt}^* \in \{0,1\}^*$ on his will, computes $\mathbf{t} = \mathsf{Hash}(\mu^* \| \mathsf{salt}^*)$, and then is devoted to the recovery of a preimage $\mathbf{s}$ for $\mathbf{t}$ under the public key $\mathcal{P}$ via the system-solving techniques.

At the heart of the attack is to solve a random system of m quadratic equations in n variables; and the state-of-the-art approach is to first take advantage of the underdeterminedness of the system by reducing to the problem of solving a system of $m' = m - 1$ equations in $n' = m - 1$ variables with the approach of Thomae and Wolf [16], and then using the hybrid WiedemannXL algorithm to solve the new system. The estimated cost of this state-of-the-art approach is

$$\min_k q^k \cdot 3 \binom{n' - k + d_{n'-k,m'}}{d_{n'-k,m'}}^2 \binom{n' - k + 2}{2} (2r^2 + r)\,, \tag{9}$$

and is *identified* as the cost of the direct attack against TUOV. Here, $d_{N,M}$ is the *operating degree* of XL, and is defined to be the smallest $d > 0$ such that the coefficient of t^d in the power series expansion of

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

is non-positive.

Note that the attacker might compute $\mathsf{Hash}(\mu \| \mathsf{salt})$ for a large number of message/salt pairs, and then solve a multi-target version of the system-solving problem. Nevertheless, our foregoing estimation is justified by the fact that there are no known algorithms that can take advantage of multiple targets (beyond the naive collision attacks).

4.2.2 Kipnis-Shamir Attack

The Kipnis-Shamir attack [8] tries to recover the subspace O from the public map $\mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$. Historically, this attack was first proposed for the case $n = 2m$, where it runs in polynomial time and demonstrates the insecurity of the *original balanced* OV scheme proposed in [1]. Moreover, it can generalized to the cases $n > 2m$, and in the literature its cost was identified as $\mathcal{O}(q^{n-2m}n^4)$, if n is even or q is odd.

However, it turns out that the foregoing formula overestimates the cost of the attack, as the following analysis indicates. First, the cost of finding a single vector in O is dominated by the cost of computing an average of q^{n-2m} characteristic polynomials of n-by-n matrices, and solving the same number of linear systems in n variables; This takes $\mathcal{O}(q^{n-2m}n^{\omega}\log(n))$ field multiplications, where ω denotes the exponent of matrix multiplication. The n^4 factor in the literature was obtained by putting $\omega = 3$. Moreover, the foregoing attack should be repeated $m = O(n)$ times so as to get a basis for O. Nevertheless, this does not contribute an m factor into the overall cost intuitively, because once a first vector in O is found, it could be fully utilized and the other vectors in O can be found more efficiently with other methods (*e.g.*, see [15]).

With this in mind, in this submission the cost of Kipnis-Shamir attack is identified as

$$q^{n-2m}n^{2.8}(2r^2 + r).$$

4.2.3 Intersection Attack

The intersection attack tries to simultaneously find k vectors in the oil space $O = \left\{\mathbf{u} \in \mathbb{F}_q^n \,\middle|\, \mathcal{P}(\mathbf{u}) = \mathbf{0}_m\right\}$, by solving a system of quadratic equations on some target vectors in the $\mathbb{F}_q$-subspace $\cap_{i=1}^{k}\mathbf{M}_i O$ determined by the matrices $\mathbf{M}_i$. As shown in [15], when $n < \frac{2k-1}{k-1}m$, the foregoing intersection subspace is of positive dimension.

The intersection attacker is parameterized by k, anmd most of its running time is devoted to solving a random system of

$$M = \binom{k+1}{2}m - 2\binom{k}{2}$$

equations in $N = kn - (2k-1)m$ variables. When k gets larger, the dimension of the foregoing intersection subspace decreases. In particular, when a larger k is chosen, the intersection subspace cannot be guaranteed to be nontrivial (*e.g.*, for the `tuov-Ip` parameter set in this submission with $k = 3$), the attack may still work with approximate probability $1/(q-1)$. Nevertheless, by running this intersection attacker multiple times, the attacker with a larger k may still outperform that with a smaller k.

4.2.4 MinRank Attack

In the *MinRank attack*, the attacker tries to find a linear combination of the public polynomials of minimal rank [17, 18].. And the *MinRank problem* can be formulated as: given the m matrices $\mathbf{P}_1, ..., \mathbf{P}_m \in \mathbb{F}_q^{n\times n}$ representing the quadratic polynomials $p_1, ..., p_m$ in the public key $\mathcal{P}$, find a linear combination $\mathbf{Q} = \sum c_i \cdot \mathbf{P}_i$ with rank no more than r. Historically, there exist many different approaches to solve the MinRank problem, including the linera algebra approach, the Kipnis-Shamir method, and the Minors Modeling method.

Let $\mathbf{A}_i$ be the submatrix of $\mathbf{P}_i$ which consists of the last $k = \left\lceil \frac{n-m}{m} \right\rceil$ rows. Then $\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \in \mathbb{F}_q^{km\times n}$ has rank no more than $n - m$. In this way, the MinRank attacker can be used to recover the secret key of TUOV, with cost

The complexity of MinRank attack on TUOV is

$$\mathcal{O}\left(\left(\left\lceil \frac{n-m}{m} \right\rceil (n-m)\binom{n'}{n-m}\right)^2 \cdot \left\lceil \frac{n-m}{m} \right\rceil (n-m)(n-m+1)\right),$$

where the positive integer $n' \in \{n-m+1, ..., n\}$ satisfies

$$m\binom{n'}{n-m+1} \geq (n-m)\binom{n'}{n-m} - 1.$$

Although this attack works for TUOV scheme, in regard to our four sets of recommended parameters, its cost estimate is much larger than those of the other attackers.

4.2.5 Quantum Attacks

All the known quantum attacks against TUOV are obtained by speeding some part of a classical attack up with Grover's algorithm. Therefore, they outperform the classical attacks by at most a square root factor, and they do not threaten our security claims.

5 Implementations and Performance

In this section, we introduce the implementation details of TUOV variants on both reference and AVX2 implementations. We also present the benchmark results on the AVX2 optimization, in order to showcase the practical strengths of TUOV.

5.1 Reference Implementation

We use the following two finite fields in our TUOV implementations:

- $\mathbb{F}_{256} := \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$;
- $\mathbb{F}_{16} := \mathbb{F}_2[x]/(x^4 + x + 1)$.

Every element in $\mathbb{F}_{256}$ could be represented by a polynomial over $\mathbb{F}_2$ with degree no more than 7, and hence could be packed into one byte precisely: its most significant bit corresponds to the coefficient of x^7, and the constant corresponds to the least significant bit, *etc.* The vector of $\mathbb{F}_{256}^{\ell}$ is represented as an ℓ-byte string. The first element of the vector is the first byte of the string.

For $\mathbb{F}_{16}$, every element could also be represented by a polynomial over $\mathbb{F}_2$ with degree no more than 3, and hence we pack every two field elements into one byte precisely: the first element is in the least significant nibble, and the most significant bit of each nibble corresponds to x^3. A vector of $\mathbb{F}_{16}^{\ell}$ is represented as an $\lceil \ell/2 \rceil$-byte string. The first element of the vector is in the least significant nibble of the first byte of the string. The most significant nibble in the last byte is zero-padded if ℓ is odd.

5.1.1 Data Layout of Keys and Signatures

In this section, we define how keys and signatures in TUOV are translated into byte strings that can be processed by computers.

Encoding of upk. Recall that the uncompressed public key

$$\mathsf{upk} = \left[\begin{array}{cccccc} \overline{\mathbf{P}}^{(1)} & \overline{\mathbf{P}}^{(2)} & \overline{\mathbf{P}}^{(3)} & \overline{\mathbf{P}}^{(5)} & \overline{\mathbf{P}}^{(6)} & \overline{\mathbf{P}}^{(9)} \end{array}\right] = \left[\begin{array}{cccccc} \overline{\mathbf{P}}_1^{(1)} & \overline{\mathbf{P}}_1^{(2)} & \overline{\mathbf{P}}_1^{(3)} & \overline{\mathbf{P}}_1^{(5)} & \overline{\mathbf{P}}_1^{(6)} & \overline{\mathbf{P}}_1^{(9)} \\ \overline{\mathbf{P}}_2^{(1)} & \overline{\mathbf{P}}_2^{(2)} & \overline{\mathbf{P}}_2^{(3)} & \overline{\mathbf{P}}_2^{(5)} & \overline{\mathbf{P}}_2^{(6)} & \overline{\mathbf{P}}_2^{(9)} \end{array}\right].$$

In implementation, a flow of bytes reformulates upk into a vector such that $\{\overline{\mathbf{P}}_i^{(1)}, \overline{\mathbf{P}}_i^{(2)}, \overline{\mathbf{P}}_i^{(3)}, \overline{\mathbf{P}}_i^{(5)}, \overline{\mathbf{P}}_i^{(6)}, \overline{\mathbf{P}}_i^{(9)}\}(i \in \{1,2\})$ are stored in a specific manner. First, $\left[\begin{array}{c} \overline{\mathbf{P}}_1^{(1)} \\ \overline{\mathbf{P}}_2^{(1)} \end{array}\right]$ is stored as a column-major $m \times \frac{(n-m)(n-m+1)}{2}$ Macaulay matrix. And then a column-major $m \times (n-m)m$ Macaulay matrix $\begin{bmatrix} \overline{\mathbf{F}}_1^{(2)} & \overline{\mathbf{F}}_1^{(3)} \\ \overline{\mathbf{F}}_2^{(2)} & \overline{\mathbf{F}}_2^{(3)} \end{bmatrix}$ is stored sequentially. Finally, $\begin{bmatrix} \overline{\mathbf{F}}_1^{(5)} & \overline{\mathbf{F}}_1^{(6)} & \overline{\mathbf{F}}_1^{(9)} \\ \overline{\mathbf{F}}_2^{(5)} & \overline{\mathbf{F}}_2^{(6)} & \overline{\mathbf{F}}_2^{(9)} \end{bmatrix}$ is stored as a column-major $m \times \frac{m(m-1)}{2}$ Macaulay matrix. This vector requires $\lceil m \cdot \frac{n(n+1)}{2} \cdot \frac{\lceil \log_2 q \rceil}{8} \rceil$ bytes for storage.

Encoding of usk. Recall that the uncompressed secret key

$$\mathsf{usk} = (\mathsf{seed}_{\mathsf{sk}}, \mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}, \{\overline{\mathbf{F}}_j^{(1)}, \overline{\mathbf{F}}_j^{(2)}, \overline{\mathbf{F}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{F}}_2^{(5)}, \overline{\mathbf{F}}_2^{(6)}\}).$$

In implementation, a flow of bytes reformulates the parts of usk into three vectors $\{\mathsf{seed}_{\mathsf{sk}}, \mathsf{ST}, \mathsf{F}\}$.

- $\mathsf{seed}_{\mathsf{sk}}$ is a random chosen string of $\lceil \mathsf{sk\_seed\_len}/8 \rceil$ bytes, *i.e.*, always 32 bytes in our implementations.
- ST stores $\mathbf{S}, \mathbf{T}^{(1)}, \mathbf{T}^{(3)}, \mathbf{T}^{(4)}$ in order. Here $\mathbf{S}$ is a $m_1 \times (m - m_1)$ matrix, $\mathbf{T}^{(1)}$ is a $(n-m) \times 1$ matrix, $\mathbf{T}^{(3)}$ a $1 \times (m-1)$ one while $\mathbf{T}^{(4)}$ is a $(n-m) \times (m-1)$ one. They are formatted as column-major matrices respectively. Notice that the length of $\mathbf{T}^{(3)}$ is odd, then in $\mathbb{F}_{16}$, 4 zero bits are padded. ST requires
$$\lceil (m_1(m - m_1) + n - 1 + (n-m)(m-1)) \frac{\lceil \log_2 q \rceil}{8} \rceil$$
bytes for storage in total.
- F stores $\{\overline{\mathbf{F}}_j^{(1)}, \overline{\mathbf{F}}_j^{(2)}, \overline{\mathbf{F}}_j^{(3)}\}_{j=1,2} \cup \{\overline{\mathbf{F}}_2^{(5)}, \overline{\mathbf{F}}_2^{(6)}\}$ in a specific manner. Firstly, $\begin{bmatrix} \overline{\mathbf{F}}_1^{(1)} \\ \overline{\mathbf{F}}_2^{(1)} \end{bmatrix}$ is stored as a column-major $m \times \frac{(n-m+1)(n-m)}{2}$ Macaulay matrix. Then is $\begin{bmatrix} \overline{\mathbf{F}}_1^{(2)} & \overline{\mathbf{F}}_1^{(3)} \\ \overline{\mathbf{F}}_2^{(2)} & \overline{\mathbf{F}}_2^{(3)} \end{bmatrix}$ as a column-major $m \times (n-m)m$ Macaulay matrix. Finally, $\begin{bmatrix} \overline{\mathbf{F}}_1^{(5)} & \overline{\mathbf{F}}_1^{(6)} \end{bmatrix}$ is stored as a column-major $(m - m_1) \times m$ Macaulay matrix. F requires
$$\lceil (m \cdot \frac{(n-m)(n-m+1)}{2} + (n-m)m^2 + (m-m_1)m) \cdot \frac{\lceil \log_2 q \rceil}{8} \rceil$$
bytes for storage in total.

Encodings of cpk. Recall that compressed public key
$$\mathsf{cpk} = (\mathsf{seed}_{\mathsf{pk}}, \{\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}, \overline{\mathbf{P}}_2^{(9)}\}).$$

In implementation, a flow of bytes reformulates cpk into four parts. First part is $\mathsf{seed}_{\mathsf{pk}}$, a random chosen string of $\lceil \mathsf{sk\_seed\_len}/8 \rceil$ bytes, *i.e.*, always 16 bytes in our implementations. Then $\overline{\mathbf{P}}_1^{(5)}$ is stored as a column-major $m_1 \times 1$ Macaulay matrix sequentially, following a $m_1 \times m_1$ Macaulay matrix $\overline{\mathbf{P}}_1^{(6)}$. Finally, $\begin{bmatrix} \overline{\mathbf{P}}_9^{(1)} \\ \overline{\mathbf{P}}_9^{(2)} \end{bmatrix}$ is stored as a column-major $m \times \frac{(m-1)(m-1)}{2}$ Macaulay matrix. The latter three parts relating to $\{\overline{\mathbf{P}}_1^{(5)}, \overline{\mathbf{P}}_1^{(6)}, \overline{\mathbf{P}}_1^{(9)}, \overline{\mathbf{P}}_2^{(9)}\}$ requires
$$\lceil (m \cdot \frac{(n-m)(n-m+1)}{2} + (n-m)m^2 + (m-m_1)m) \cdot \frac{\lceil \log_2 q \rceil}{8} \rceil$$
bytes for storage in total.

Encodings of csk. The compressed secret key $\mathsf{csk} = (\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}})$ is separated into two vectors containing $\lceil \mathsf{pk\_seed\_len}/8 \rceil$, $\lceil \mathsf{sk\_seed\_len}/8 \rceil$ bytes respectively.

Encodings of signature. The byte string of $\sigma \in \mathbb{F}_q^n$ is $\lceil n \frac{\lceil \log_2 q \rceil}{8} \rceil$ bytes.

5.1.2 Common Implementation Techniques

In this section, we describe our implementation techniques that are shared among platforms for solving linear equations as well as verification.

Solving linear equations.

Algorithm 1 Constant-time linear equation solving using Gaussian elimination directly

```
Input: Linear equation $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{F}^{m \times \ell}$ $(\ell \geq m)$
Output: Solution $\mathbf{x} \in \mathbf{F}^{\ell}$ or $\perp$
 1: for $i = m$ upto $\ell - 1$ do
 2:     $x_i \xleftarrow{\$} \mathbb{F}_q$
 3: for $i = 0$ upto $m - 1$ do
 4:     for $j = m$ upto $\ell - 1$ do
 5:         $b_i := b_i + a_{i,j} \cdot x_j$
 6: $\mathbf{A}' =$ (first $m$ columns of $\mathbf{A} \mid \mathbf{b}$) $\in \mathbb{F}^{m \times (m+1)}$
 7: for $i = 0$ upto $m - 1$ do
 8:     for $j = i + 1$ upto $m - 1$ do
 9:         if $a'_{i,i} = 0$ then
10:             for $k = i$ upto $m$ do
11:                 $a'_{i,k} := a'_{i,k} + 1 \cdot a'_{j,k}$
12:         else
13:             for $k = i$ upto $m$ do
14:                 $a'_{i,k} := a'_{i,k} + 0 \cdot a'_{j,k}$
15:     if $a'_{i,i} = 0$ then return $\perp$
16:     $p := a'_{i,i}$
17:     for $k = i$ upto $m$ do
18:         $a'_{i,k} := a'_{i,k} \cdot p^{-1}$
19:     for $j = i + 1$ upto $m - 1$ do
20:         for $k = i$ upto $m$ do
21:             $a'_{j,k} := a'_{j,k} + a'_{j,i} \cdot a'_{i,k}$
22: for $i = m - 1$ downto $0$ do
23:     $x_i := a'_{i,m}$
24:     for $j = i + 1$ upto $m - 1$ do
25:         $x_i := x_i + a'_{i,j} \cdot x_j$
    return $\mathbf{x} = (x_0, \ldots, x_{\ell-1})$
```

TUOV signing requires solving two systems of linear equations: one is $\mathbf{M}^{\top}\mathbf{v} = \mathbf{e_1}$ with $\mathbf{M}^{\top} \in \mathbb{F}_q^{m \times v}$, the other is $\mathbf{Lx} = \tilde{\mathbf{t}} - \mathbf{y}$ with a square matrix $\mathbf{L} \in \mathbb{F}_q^{(m-1)\times(m-1)}$. We outline the both approaches according to Algorithm 1. At first (line 1-2), we sample the last $\ell - m$ entries of $\mathbf{x}$ in $\mathbb{F}_q^{\ell - m}$ as the free variables in the case the linear system has a fundamental set of solutions. Secondly, we add them to the right-hand constant side to reduce the system to m variables (line 3-5, be aware that addition is equivalent to subtraction in the fields of characteristic 2). Next, we conditionally add all following rows in order to make sure the pivoting element $a'_{i,i}$ is non-zero and perform the operations in constant time (line 8-14). In case it is still zero, we return $\perp$ (line 15) as the matrix is not invertible or the newly obtained system of linear equations has no unique solution. Subsequently, we multiply the current row by the inverse of the current pivoting element (line 16-18). We then add multiples of that row to the lower part of the remaining matrix to obtain a row echelon form (line 19-21). Finally, we back-substitute the variables into the system of equations to acquire the solutions directly (line 22-25).

Directly solving a system of linear equations of m equations and v variables as in Algorithm 1 requires $vm + \frac{m^3}{3}$ multiplications approximately.

Restricting the number of conditional additions. In line 8-14 of Algorithm 1, we perform a large number of conditional additions to insure constant time. As the additions will be meaningless once the the pivoting element has become nonzero, we can restrict the number of additions to a smaller number in order to obtain better performance. In practice, we choose to add at most 15 (*resp.* 7) subsequent rows in $\mathbb{F}_{16}$ (*resp.* $\mathbb{F}_{256}$).

Using accumulators to reduce multiplications. In the verification procedure of TUOV, we need to evaluate and check whether $[\mathbf{s}^\top \mathbf{P}_k \mathbf{s}]_{k\in[m]} == \mathbf{t}$. This is identical to the one in Rainbow and may involve a very large number of multiplication. As Chou, Kannwischer, and Yang has introduced some techniques to greatly reduce the number of multiplication, we avail ourselves of these ingenious techniques to accelerate verification: As we have to perform many multiplications like $p_{i,j} \cdot s_i s_j$ when computing one $\mathbf{s}^\mathsf{T}\mathbf{P}_i\mathbf{s}$, we can sort these $p_{i,j}$ into 15 accumulators according to the multiplicands $s_i s_j \in \mathbb{F}_{16}^\times$ to defer the multiplications until the end. While dealing with $\mathbb{F}_{256}$, we use $15 \times 2 = 30$ accumulators instead of 255: one 15 for the four most significant bits and the other 15 for the four least significant bits. In both cases, the verification procedure can be accelerated significantly.

Additionally, recall that we format the $\mathsf{upk} = \{\mathbf{P}_k\}_{k\in[m]}$ as a column-major Macaulay matrix $\overline{\mathbf{P}}$ and each of the column is representing all (i,j)-th entries in $\mathbf{P}_k$ sequentially. Hence we can skip the evaluation of the column related to $p_{i,j}$'s when $s_i s_j = 0$. A fraction of $(2q-1)/q^2 \approx 2/q$ among all $s_i s_j$ is expected to be zero. This can lead to a great speed-up when dealing with $\mathbb{F}_{16}$.

Skipping parts of sampling when acquiring upk. We use compressed public keys when using pkc or $\mathsf{pkc} + \mathsf{skc}$ variant of TUOV. In both cases, we evaluate $\mathsf{Expand}_{\mathbf{P}}(\mathsf{seed}_{\mathsf{pk}})$ to pseudo-randomly sample the $\mathbf{P}_i^{(5)}, \mathbf{P}_i^{(6)} (i \in [m_1])$ and $\mathbf{P}_i^{(9)} (i \in [m])$ matrices in the verification procedure. As discussed before, if some variables in the signature is zero, *i.e.*, some $s_i s_j = 0$, then the column related to (i,j)-th entry in the Macaualy matrix form is useless when evaluating the result. Hence, we can skip sampling the corresponding elements in the $\mathbf{P}_i^{(5)}, \mathbf{P}_i^{(6)} (i \in [m_1])$ and $\mathbf{P}_i^{(9)} (i \in [m])$ by using a PRNG that can sample one position at a time sequentially instead of sampling all positions in one call. In practice, we choose aes128ctr as the PRNG. Note that this technique also gives rise to a significant speed boost especially when working in $\mathbb{F}_{16}$.

5.2 AVX2 Optimized Implemention

In this section, we present the AVX2 optimization of TUOV on the Intel Skylake architecture, which is designated as the reference platform in NIST PQC standardization [42]. We focus on the AVX2 instruction set, which is considered the most useful instruction set for its availability on modern x86 platforms.

The C source code is compiled using gcc version 9.4.0 (-1ubuntu1 20.04.1), and the performance results are measured on a laptop with an Intel Core i5-10210U CPU 1.60GHz (Skylake) with turbo boost and hyper-threading disabled.

Table 3 presents the performance comparison of our AVX2 implementations with those of Dilithium [19]. In the table, we combine the results for Sign() from both the `classic` and `pkc` versions, and results for **Verify**() from both the `pkc` and `pkc+skc` versions to indicate the same implementations. From the comparisons shown in Table 3, we observe that: 1) `tuov-Ip` has the fastest signing performance; 2) `tuov-Is` has the fastest verification performance despite having a larger public key size than `tuov-Ip`. This can be attributed to the fact that `tuov-Ip` uses more XOR operations for the two accumulators while evaluating $\mathbb{F}_{256}$ public polynomials (refer to Section verify); 3) When verifying with

Table 3: Benchmarking results of AVX2 implementations. Numbers are the median CPU cycles of 10000 executions each.

	Optimized Implementations (AVX2)		
Schemes	KeyGen	Sign	Verify
`tuov-Ip`	10,682,834	220,792	127,722
`tuov-Ip-pkc`			491,120
`tuov-Ip-pkc+skc`	6,617,102	6,698,588	
`tuov-Is`	32,007,930	272,394	103,746
`tuov-Is-pkc`			570,194
`tuov-Is-pkc+skc`	15,635,380	21,534,990	
Dilithium-II	113,316	272,332	123,916
`tuov-III`	57,322,074	608,604	442,770
`tuov-III-pkc`			1,914,056
`tuov-III-pkc+skc`	33,336,974	33,409,538	
Dilithium-III	197,026	448,172	199,656
`tuov-V`	139,948,218	1,133,958	786,450
`tuov-V-pkc`			4,520,748
`tuov-V-pkc+skc`	85,778,546	74,923,822	
Dilithium-V	303,434	551,760	313,096

compressed keys, the execution time is dominated by the computation of $\mathsf{Expand}_{\mathsf{P}}$; 4) During signing with compressed secret keys, the primary computational load is expended on the process of $\mathsf{Expand}_{\mathsf{sk}}$.

6 Summary: Strengths and Limitations of TUOV

This section encapsulates the strengths and constraints of the Triangular Unbalanced Oil and Vinegar (TUOV) digital signature scheme.

In comparison to other post-quantum digital signature schemes, the principal advantages of the TUOV signature scheme include:

Efficiency. The signature generation procedure of TUOV consists of rudimentary linear algebra operations such as matrix-vector multiplication and resolution of linear systems over small finite fields. Consequently, the TUOV scheme exhibits excellent efficiency and stands as one of the swiftest signature schemes available.

Concise signatures. The signatures produced by the TUOV signature scheme are concise, spanning only a few hundred bits. This makes them significantly shorter than those of RSA and other post-quantum signature schemes.

Modest computational demands. Given that TUOV solely requires elementary linear algebra operations over a small finite field, it can be implemented efficiently on cost-effective devices, eliminating the need for a dedicated cryptographic coprocessor.

Security. While we can only provide with a reduction from the MQ problem to our TUOV problem, we hold steadfast confidence in our scheme's security. No effective attack against TUOV has been discovered since its inception, despite intensive cryptanalysis efforts. Furthermore, we observe that, unlike some other post-quantum schemes, the theoretical complexities of known attacks against TUOV correspond impeccably with empirical data. Therefore, we remain assured of the TUOV signature scheme's overall security.

Simplicity. The design of the TUOV scheme is profoundly simple, requiring minimal algebraic knowledge for understanding and implementation. This straightforwardness suggests a paucity of structures within the scheme that could be exploited for attacks.

On the flip side, TUOV's primary drawback lies in the large size of its public keys. For security levels surpassing 128-bit, TUOV's public key sizes are substantially larger than those of traditional schemes such as RSA, ECC, and certain other post-quantum schemes. Nonetheless, with the ever-increasing memory capacities of even medium-sized devices, such as smartphones, we believe this will not pose a significant hurdle. Moreover, we propose several variants of TUOV to cater to a diverse array of practical requirements.

References

[1] Jacques Patarin. The Oil and Vinegar signature scheme. In *Dagstuhl Workshop on Cryptography*, September 1997.

[2] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

[3] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

[4] Andreas Hulsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kolbl, Tanja Lange, Martin M Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.

[5] 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. *Cryptology ePrint Archive*, 2023.

[6] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[7] Jacques Patarin. Cryptanalysis of the Matsumoto and Imai public key scheme of eurocrypt'88. pages 248–261, 1995.

[8] Aviad Kipnis and Adi Shamir. Cryptanalysis of the Oil & Vinegar signature scheme. pages 257–266, 1998.

[9] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. pages 206–222, 1999.

[10] Albrecht Petzoldt, Stanislav Bulygin, and Johannes Buchmann. CyclicRainbow - a multivariate signature scheme with a partially cyclic public key. pages 33–48, 2010.

[11] Jingtai Ding, Ming-Shing Chen, Matthias Julias Kannwischer, Albrecht Petzoldt, Jacques Patarin, Dieter Schmidt, and Bo-Yin Yang. Rainbow 3rd round submission, nist submission document and technical report, October 2020.

[12] FIPS PUB 202 – SHA-3 standard: Permutation-based hash and extendable-output functions, 2015. `https://doi.org/10.6028/NIST.FIPS.202`.

[13] FIPS PUB 197 – advanced encryption standard (AES), 2001. `https://doi.org/10.6028/NIST.FIPS.197`.

[14] Shay Gueron. Intel advanced encryption standard (aes) new instructions set, 2010. `https://www.intel.com.bo/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf`.

[15] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. pages 348–373, 2021.

[16] Enrico Thomae and Christopher Wolf. Solving underdetermined systems of multivariate quadratic equations revisited. pages 156–171, 2012.

[17] Olivier Billet and Henri Gilbert. Cryptanalysis of rainbow. In *Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5*, pages 336–347. Springer, 2006.

[18] Louis Goubin and Nicolas T Courtois. Cryptanalysis of the ttm cryptosystem. In *Advances in CryptologyASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings 6*, pages 44–57. Springer, 2000.

[19] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`.