TUOV
Triangular Unbalanced Oil and Vinegar

Boru Gong

tuovsig@gmail.com, Jintai.Ding@gmail.com

September 4, 2023

Outline

Multivariate Signature Scheme

- **Public Key**:

$$\mathcal{P}(x_1, \cdots, x_n) = (p_1(x_1, \cdots, x_n), \cdots, p_m(x_1, \cdots, x_n)),$$

where each p_i is a multivariate polynomial over $\mathbb{F}_q$.

- **Private Key**: a way to compute $\mathcal{P}^{-1}$.
- **Signing a hash of a document**:

$$(x_1, \cdots, x_n) \in \mathcal{P}^{-1}(y_1, \cdots, y_m).$$

- **Verifying**:

$$(y_1, \cdots, y_m) \stackrel{?}{=} \mathcal{P}(x_1, \cdots, x_n).$$

Multivariate Signature Schemes

- The public key $\mathcal{P}(x_1, \cdots, x_n)$ should be *almost surjective*;
  - $n \geq m$ is necessary.
- The signing and verification should be *efficient*;
- Key sizes should be as *small* as possible.

Theoretical Foundation

Direct attack is to solve the set of equations:

$$\mathcal{P}(x_1, ..., x_n) = (y'_1, ..., y'_m).$$

Quadratic Constructions

- **Efficiency considerations** lead to mainly *quadratic* constructions.

$$p_\ell(x_1, \dots, x_n) = \sum_{i,j} \alpha_{\ell ij} x_i x_j + \sum_i \beta_{\ell i} x_i + \gamma_\ell.$$

- **Mathematical structure consideration**: any set of *higher* degree polynomial equations can be reduced to a set of quadratic equations.
  - For instanve, $x_1x_2x_3 = 5$ is equivalent to

$$\begin{aligned} x_1x_2 - y &= 0 \\ yx_3 &= 5. \end{aligned}$$

What is MQ Problem

MQ Problem

Given: a multivariate quadratic equation system with m equations and n variables over $\mathbb{F}_q$.
Find: a solution over $\mathbb{F}_q$.

Hardness of MQ Problem

Theorem

1. MQ problem is NP-complete. [Garey-Johnson 1979]
2. MQ problem can be solved in polynomial time,
   when: $m \leq n^2/2$ [Shamir 1999],
   when: $n \geq m^2/2$ and $\mathrm{char}(\mathbb{F}_q) = 2$ [Miura-Hashimoto-Takagi 2013].

Moreover, it is *believed* $\exists 0 < \varepsilon < 1/2$ s.t. MQ problem is hard when $n = \alpha m^2$ and $\varepsilon < \alpha < 1/2$.

(Unbalanced) Oil and Vinegar Scheme

- The original *balanced* version was introduced by Jacques Patarin in 1997.
  - Inspired by **linearization attack** to Matsumoto-Imai cryptosystem.
  - $n = 2m$.
- Kipnis and Shamir proposed an attack which breaks this balanced OV scheme in 1998.
- Kipnis, Patarin and Goubin proposed a modified scheme called *Unbalanced* Oil and Vinegar (UOV) signature scheme in 1999.
  - $n > 2m$.

(Unbalanced) Oil and Vinegar Scheme

- Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements;
- v: the number of *vinegar* variables;
  $m = o$: the number of *oil* variables;
  $n := v + m$: the number of variables.
- $V = \{1, \ldots, v\}$, $O = \{v + 1, \ldots, n\}$.
  We denote the variables x_i $(i \in V)$ as **Vinegar variables**,
  $x_{v+1}, \ldots, x_n$ as **Oil variables**.

(Unbalanced) Oil and Vinegar Scheme

OV-polynomial

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

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

The homogeneous quadratic part of f can be uniquely represented in a upper-triangular quadratic form: $\begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{0}_{m\times m} \end{bmatrix}$.

Note that there is no "Oil $\times$ Oil" part in an OV-polynomial.

Key Generation

UOV Central Map

$\mathcal{F} = (f_1, \ldots, f_m)$, where each f_k is an

$$(n, m)\text{-OV-polynomial}, \;\; k = 1, \ldots, m.$$

- **Private Key** is $= (\mathcal{T}, \mathcal{F})$, where
  - $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ is a UOV central map.
    - $\mathcal{F} = (f_1, ..., f_m)$.
  - $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ is an affine map;
- **Public Key** is $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$.

Signature Generation

$\mathbf{z} \leftarrow \mathsf{Sign}(\mu)$:

1. Use a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}^m$ to compute a digest $\mathbf{w} = \mathcal{H}(\mu)$.
2. Find a pre-image $\mathbf{y} \in \mathbb{F}^n$ of $\mathbf{w}$ under the central map $\mathcal{F}$.
3. Compute the signature $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

How to find $\mathcal{F}^{-1}$

Inversion of $\mathcal{F}^{-1}$ is efficient indeed:

1. Fix values for vinegar variables $x'_1, \cdots, x'_v$.
2. $f_k = \sum \alpha_{i,j}^{(k)} x_i x'_j + \sum \alpha_{i,j}^{(k)} x'_i x'_j + \sum \beta_i^{(k)} x_i + \sum \beta_i^{(k)} x'_i + \gamma^{(k)}$
3. $\mathcal{F}(x'_1, \cdots, x'_v, \cdots) : \mathbb{F}_q^m \to \mathbb{F}_q^m$ is a linear system in oil variables $x_{v+1}, \cdots, x_n$.

Signature Verification

Signature verification is fast:

- Check whether $\mathcal{H}(d) = \mathcal{P}(\mathbf{z})$. If so, then the signature $\mathbf{z}$ is accepted, otherwise rejected.

Attacks on UOV

- Kipnis-Shamir attack (*i.e.*, UOV attack)
- Reconciliation attack
- Collision attack
- Direct attack
- Intersection attack
- MinRank attack
- Quantum attack

UOV: Parameter Sets

	NIST S.L.	n	m	q	|epk| (bytes)	|esk| (bytes)	|cpk| (bytes)	|csk| (bytes)	signature (bytes)
uov-Ip	1	112	44	256	278 432	237 896	43 576	48	128
uov-Is	1	160	64	16	412 160	348 704	66 576	48	96
uov-III	3	184	72	256	1 225 440	1 044 320	189 232	48	200
uov-V	5	244	96	256	2 869 440	2 436 704	446 992	48	260

Figure: Recommended parameter sets and and the corresponding key/signature sizes for UOV.

UOV: Performances

<table>
<tr><th rowspan="2"></th><th colspan="3">Haswell</th><th colspan="3">Skylake</th></tr>
<tr><th>KeyGen</th><th>Sign</th><th>Verify</th><th>KeyGen</th><th>Sign</th><th>Verify</th></tr>
<tr><td>uov-Ip-classic</td><td>3 311 188</td><td rowspan="2">116 624</td><td>82 668</td><td>2 903 434</td><td rowspan="2">105 324</td><td>90 336</td></tr>
<tr><td>uov-Ip-pkc</td><td>3 393 872</td><td rowspan="2">311 720</td><td>2 858 724</td><td rowspan="2">224 006</td></tr>
<tr><td>uov-Ip-pkc+skc</td><td>3 287 336</td><td>2 251 440</td><td>2 848 774</td><td>1 876 442</td></tr>
<tr><td>uov-Is-classic</td><td>4 945 376</td><td rowspan="2">123 376</td><td>60 832</td><td>4 332 050</td><td rowspan="2">109 314</td><td>58 274</td></tr>
<tr><td>uov-Is-pkc</td><td>5 002 756</td><td rowspan="2">398 596</td><td>4 376 338</td><td rowspan="2">276 520</td></tr>
<tr><td>uov-Is-pkc+skc</td><td>5 448 272</td><td>3 042 756</td><td>4 450 838</td><td>2 473 254</td></tr>
<tr><td>uov-III-classic</td><td>22 046 680</td><td rowspan="2">346 424</td><td>275 216</td><td>17 603 360</td><td rowspan="2">299 316</td><td>241 588</td></tr>
<tr><td>uov-III-pkc</td><td>22 389 144</td><td rowspan="2">1 280 160</td><td>17 534 058</td><td rowspan="2">917 402</td></tr>
<tr><td>uov-III-pkc+skc</td><td>21 779 704</td><td>11 381 092</td><td>17 157 802</td><td>9 965 110</td></tr>
<tr><td>uov-V-classic</td><td>58 162 124</td><td rowspan="2">690 752</td><td>514 100</td><td>48 480 444</td><td rowspan="2">591 812</td><td>470 886</td></tr>
<tr><td>uov-V-pkc</td><td>57 315 504</td><td rowspan="2">2 842 416</td><td>46 656 796</td><td rowspan="2">2 032 992</td></tr>
<tr><td>uov-V-pkc+skc</td><td>57 306 980</td><td>26 021 784</td><td>45 492 216</td><td>22 992 816</td></tr>
<tr><td>Dilithium 2† [28]</td><td>97 621*</td><td>281 078*</td><td>108 711*</td><td>70 548</td><td>194 892</td><td>72 633</td></tr>
<tr><td>Falcon-512 [44]</td><td>19 189 801*</td><td>792 360*</td><td>103 281*</td><td>26 604 000</td><td>948 132</td><td>81 036</td></tr>
<tr><td>SPHINCS+‡ [25]</td><td>1 334 220</td><td>33 651 546</td><td>2 150 290</td><td>1 510 712*</td><td>50 084 397*</td><td>2 254 495*</td></tr>
</table>

† Security level II. ‡ Sphincs+-SHA2-128f-simple. * Data from SUPERCOP [20].

Figure: Benchmarking results of AVX2 implementations. Numbers are the median CPU cycles of 1000 executions each.

Why we do TUOV?

- The hardness of the UOV scheme relies on the **UOV assumption**, *i.e.*, it is hard to find a pre-image of $\mathcal{P}$.
- On the one hand, it is know that

$$\text{find a pre-image of } \mathcal{P} \ \leq \ \text{solve MQ problem.}$$

- On the other hand, it is not know whether

$$\text{find a pre-image of } \mathcal{P} \overset{?}{=} \text{solve MQ problem.}$$

- a large part of coefficients in $\mathcal{F}$ are zeroes, which makes $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ failed to be proved as random as MQ map.

Why TUOV?

Our solution:

- add some nonzero parts in $\mathcal{F}$; and
- keep $\mathcal{F}$ efficiently invertible.

Our result: the *Triangular* Unbalanced Oil and Vinegar (TUOV) scheme.

Triangular Map

Definition

The **T**riangular in the name TUOV refers to a triangular map (or, *de Jonquiére* map) as

$$\begin{aligned}\mathcal{J} : \mathbb{F}_q^n &\to \mathbb{F}_q^n, \\ \mathbf{x} &\mapsto (x_1, x_2 + g_2(x_1), \cdots, x_n + g_n(x_1, \ldots, x_{n-1}))\end{aligned}$$

where g_i is a polynomial over $\mathbb{F}_q$.

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

TUOV notations

- parameters: let m, v, o_1 be integers and the number of variables is given by $n = v + o_1 + (m - o_1)$.
- index sets:
  $V = \{1, \ldots, v\}$,
  $O_1 = \{v + 1, \ldots, v + o_1\}$,
  $O_2 = \{v + o_1 + 1, \cdots, n\}$.
- variables: denote x_i $(i \in V)$ as Vinegar variables, and $x_{v+1}, \ldots, x_n$ $(O_1 \cup O_2)$ as **Oil variables**.

TOV-polynomial

Definition

For $d \geq 1$, an (n, m, d)-**TOV-polynomial** f over $\mathbb{F} = \mathbb{F}_q$ is defined as

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

The homogeneous quadratic part of f:

$$\begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \mathbf{A}^{(3)} \\ \mathbf{0} & \mathbf{A}^{(5)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

(n, m, d)-TOV-polynomial adds an "$O_1 \times O_1$ $(o_1 = d)$" part.

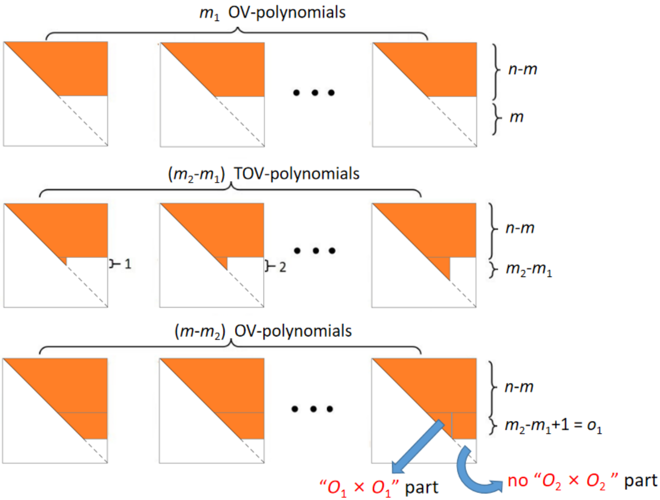
TUOV Central Map

Definition

A *TUOV central map* with parameters (n, m, m_1, m_2, q) is $\mathcal{F} = (f_1, \ldots, f_m)$, where f_k is

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

TUOV Central Map



Key Generation

- **Private Key**: $(\mathcal{S}, \mathcal{F}, \mathcal{T})$, where
  - $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$, $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ are affine maps;
  - $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ is a TUOV central map.
- **Public Key**: $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$.

Signature Generation

$\mathbf{z} \leftarrow \mathrm{Sign}(\mu)$:

1. Use a hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{F}^m$ to compute a digest $\mathbf{w} = \mathcal{H}(\mu)$.
2. Compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^m$.
3. Find a pre-image $\mathbf{y} \in \mathbb{F}^n$ of $\mathbf{x}$ under the central map $\mathcal{F}$.
4. Compute the signature $\mathbf{z} \in \mathbb{F}^n$ by $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

How to find the pre-image under $\mathcal{F}$ (1/2)

Step 1: Use one (n, m)-OV-polynomial and all TOV-polynomials in $\mathcal{F}$:

$$\begin{array}{r} \sum_{j=v+1}^{n} L_{1,j}(x_1,\cdots,x_v)x_j + Q_1(x_1,\cdots,x_v) \\ g(x_{v+1}) + \sum_{j=v+1}^{n} L_{2,j}(x_1,\cdots,x_v)x_j + Q_2(x_1,\cdots,x_v) \\ \cdots \\ g(x_{v+1},\cdots,x_{v+o_1-1}) + \sum_{j=v+1}^{n} L_{o_1,j}(x_1,\cdots,x_v)x_j + Q_{o_1}(x_1,\cdots,x_v) \end{array}$$

Solve linear equations of x_i $(i \in V)$, so that the red parts $= 0$ and blue parts $= 1$. Substitute the solution, then we can solve x_i $(i \in O_1)$ efficiently – by triangular map.

$$\begin{array}{cccccc} L_{1,v+1} & L_{1,v+2} & \cdots & L_{1,v+o_1} & \cdots & L_{1,n} \\ L_{2,v+1} & L_{2,v+2} & \cdots & L_{2,v+o_1} & \cdots & L_{2,n} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ L_{o_1,v+1} & L_{o_1,v+2} & \cdots & L_{o_1,v+o_1} & \cdots & L_{o_1,n} \end{array}$$

How to find the pre-image under $\mathcal{F}$ (2/2)

Step 2: Use remaining OV-polynomials in $\mathcal{F}$:
Substitute x_i ($i \in V \cup O_1$), we can get x_i ($i \in O_2$) by solving linear system.

Note

Note that a (n, m, d)-TOV-polynomial adds a (n, m)-polynomial is still a (n, m, d)-TOV-polynomial, thus we can randomly add (n, m)-polynomials to (n, m, d)-TOV-polynomial to get random x_i ($i \in V$) at the beginning.

Signature Verification

- Check whether $\mathcal{H}(d) = \mathcal{P}(\mathbf{z})$. If so, then the signature $\mathbf{z}$ is accepted, otherwise rejected.

Security Proof

Definition: Hardness of MQ problem

The *MQ problem* parameterized by (n, m, q) is called (t, ε)-hard, if there exists no algorithm that, given a random MQ-map $\mathcal{M} : \mathbb{F}^n \to \mathbb{F}^m$, on input $\mathbf{y} := \mathcal{M}(\mathbf{w})$ with $\mathbf{w} \xleftarrow{\$} \mathbb{F}^n$, outputs $\mathbf{w}'$ such that $\mathcal{M}(\mathbf{w}') = \mathbf{y}$ with probability no less than ε in processing time t.

Security Proof

TUOV map

TUOV map is $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^m$ where $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$ are invertible affine transformations and $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ is a TUOV central map.

Definition: Hardness of TUOV problem

TUOV problem with params (n, m, m_1, m_2, q) is (t, ε)-hard if there exists no algorithm that, given a random TUOV map $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$, on input $\mathbf{z} = \mathcal{P}(\mathbf{w})$ with $\mathbf{w} \xleftarrow{\$} \mathbb{F}^n$, outputs $\mathbf{w}'$ such that $\mathcal{P}(\mathbf{w}') = \mathbf{z}$ with probability no less than ε in processing time t.

Security Proof

Theorem

If MQ problem with params $(n = \frac{11}{24} \cdot m^2, m, q)$ *is* (t, ε)*-hard, then TUOV problem with params* $(n = \frac{1}{2} \cdot m^2, m, m_1 = \frac{1}{2} \cdot m, m_2 = \frac{3}{4} \cdot m, q)$ *is* (t, ε)*-hard.*

Proof Sketch

- Our goal is to prove: it is highly probable that for a random MQ-map $\mathcal{M}$, there exists an invertible affine map $\mathcal{Q}$ and a TUOV central map $\mathcal{F}$ such that $\mathcal{M} = \mathcal{F} \circ \mathcal{Q}$.
- Hence, if there exists an algorithm $\mathcal{A}$ that solves TUOV problem, we can construct an algorithm $\mathcal{B}^{\mathcal{A}}$ that solves MQ-problem.
- Specifically, when taking MQ input $(\mathcal{M}, \mathbf{y})$, $\mathcal{B}^{\mathcal{A}}$ can randomly sample an $\mathcal{S}$, make query $(\mathcal{P} = \mathcal{S} \circ \mathcal{M}, \mathbf{z} = \mathcal{S}(\mathbf{y}))$ on $\mathcal{A}$ and output the $\mathbf{w}'$ returned by $\mathcal{A}$.

Proof Sketch

Without loss of generality, we consider quadratic polynomials only with their quadratic part.

- Set the matrix representation of $\mathcal{Q}^{-1}$ to be

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I}_{n-m} & \mathbf{Q}^{(2)} \\ \mathbf{0}_{m\times(n-m)} & \mathbf{I}_m \end{bmatrix}.$$

- There are $(n-m)\times m$ unknown variables.
- Recall our goal is to find $\mathcal{F}$ and $\mathcal{Q}$ such that $\mathcal{M} = \mathcal{F} \circ \mathcal{Q}$.
- Hence we want $\mathcal{M} \circ \mathcal{Q}^{-1}$ to be some TUOV central map, i.e. it satisfies some equations.

Proof Sketch

The total number of equations to solve is

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

If we pick $m_1=\frac{1}{2}m$ and $m_2=\frac{3}{4}m$, *i.e.*, $m_2-m_1=\frac{1}{4}m$, then we have roughly $\frac{11}{24}m^3$ equations. As long as the number of variables is no less than equations, *i.e.*, $n\geq\frac{11}{24}\cdot m^2$, there exists invertible linear transformation $\mathcal{Q}$ such that $\mathcal{M}\circ\mathcal{Q}^{-1}$ is TUOV central map with high probability.

Attacks on TUOV

To our knowledge, attacks against UOV is applicable to TUOV, and vice versa.

TUOV: Parameter Sets

For efficient implementation, we choose $m_1 = m_2$.

	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

Figure: Recommended parameter sets and and the corresponding key/signature sizes for TUOV.

TUOV: Performances

<table>
<tr><th rowspan="2">Schemes</th><th colspan="3">Optimized Implementations (AVX2)</th></tr>
<tr><th>KeyGen</th><th>Sign</th><th>Verify</th></tr>
<tr><td>tuov-Ip</td><td rowspan="2">10,682,834</td><td rowspan="2">220,792</td><td>127,722</td></tr>
<tr><td>tuov-Ip-pkc</td><td rowspan="2">491,120</td></tr>
<tr><td>tuov-Ip-pkc+skc</td><td>6,617,102</td><td>6,698,588</td></tr>
<tr><td>tuov-Is</td><td rowspan="2">32,007,930</td><td rowspan="2">272,394</td><td>103,746</td></tr>
<tr><td>tuov-Is-pkc</td><td rowspan="2">570,194</td></tr>
<tr><td>tuov-Is-pkc+skc</td><td>15,635,380</td><td>21,534,990</td></tr>
<tr><td>Dilithium-II</td><td>113,316</td><td>272,332</td><td>123,916</td></tr>
<tr><td>tuov-III</td><td rowspan="2">57,322,074</td><td rowspan="2">608,604</td><td>442,770</td></tr>
<tr><td>tuov-III-pkc</td><td rowspan="2">1,914,056</td></tr>
<tr><td>tuov-III-pkc+skc</td><td>33,336,974</td><td>33,409,538</td></tr>
<tr><td>Dilithium-III</td><td>197,026</td><td>448,172</td><td>199,656</td></tr>
<tr><td>tuov-V</td><td rowspan="2">139,948,218</td><td rowspan="2">1,133,958</td><td>786,450</td></tr>
<tr><td>tuov-V-pkc</td><td rowspan="2">4,520,748</td></tr>
<tr><td>tuov-V-pkc+skc</td><td>85,778,546</td><td>74,923,822</td></tr>
<tr><td>Dilithium-V</td><td>303,434</td><td>551,760</td><td>313,096</td></tr>
</table>

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

Summary

- UOV is competitive with the new NIST standards by most measures, except for public key size.
- with the triangular structure, TUOV has slower signing speed than UOV, while it gains a reduction to hard problem (MQ-problem when $n = \alpha m^2$ where $\varepsilon < \alpha < 1/2$).

Thanks and Any Questions?

tuovsig@gmail.com, Jintai.Ding@gmail.com