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

QR-UOV のマスキング実装に関する提案
Proposal for masking implementation of QR-UOV

中邑 聡史 * 金城 皓羽 * 古江 弘樹 *
Satoshi Nakamura Koha Kinjo Hiroki Furue

あらまし NIST では署名方式に関して 2022 年より構造付き格子を利用しない方式を求めて追加公募を開始しており，現在では 14 方式が第 2 ラウンドに進出している．本稿では，第 2 ラウンドに進出した方式の 1 つである QR-UOV に関して，安全な実装方式であるマスキング実装を提案する．マスキング実装は秘密情報を分散したままで演算を行う手法であり，暗号が実装されたハードウェアに物理的に干渉するサイドチャネル攻撃の対策としてしられている．既に先行して NIST の標準化方式に選ばれた格子暗号方式では様々なマスキング実装が提案されているが，QR-UOV に関しては本提案が初めてである．

キーワード 耐量子計算機暗号，QR-UOV，マスキング，サイドチャネル攻撃

1 はじめに

米国標準技術研究所（NIST）では，量子計算機の解読にも耐性を持つ次世代暗号技術として，耐量子計算機暗号の標準化を進めている [13]．2016 年に暗号化/鍵交換方式及びデジタル署名に関する公募を開始し，2022 年には 4 つの方式が先行して標準化された．署名方式に関しては 3 件が標準化方式に選定されたが，その内 2 件は構造付き格子を利用した暗号方式であったこともあり，NIST は構造付き格子以外の方式を中心に署名の再公募をかけ，再びラウンド方式で標準化選定を行っている．追加公募では 40 件の方式が第 1 ラウンドに進んでおり，2024 年 10 月にはさらに 14 件に絞られた．

多変数多項式暗号は，有限体上での多変数二次多項式系の求解問題の困難性に基づいて構成されており，耐量子計算機暗号の有力な候補と見なされている．実際に追加公募で応募された第 1 ラウンドの方式 40 件のうち多変数多項式暗号が最多の 10 件を占めていた．多変数多項式問題は NP 完全 [8] であることが証明されており，量子計算機を用いた攻撃に対して安全であると期待されている．さらに，多変数多項式暗号は，特にデジタル署名方式の構成に適性を持つことが知られている．

Unbalanced Oil and Vinegar（UOV）[10] 署名は，Kipnis らによって EUROCRYPT 1999 にて提案された多変数多項式署名であり，現在に至るまで 20 年以上にわたって安全性の保たれた方式である．また UOV は，短い署名を持ち，高速な処理が可能なことで知られている．一方で UOV には，格子ベースの方式などの他の耐量子計算機暗号の候補と比較して，公開鍵が非常に大きいという欠点が存在する．2021 年には古江らによって UOV の亜種として，quotient ring UOV（QR-UOV）[7] が ASIACRYPT2021 にて提案された．QR-UOV では剰余環を用いることで公開鍵サイズを減らすことができる方式であり，NIST の追加公募の第 2 ラウンドに進んでいる方式の 1 つである．その他に第 2 ラウンドに進んでいる多変数多項式暗号としては，UOV,MAYO[4],SNOVA[14] がある．

近年，IC カードのような暗号モジュールに対して行われる実装攻撃が脅威となっている．実装攻撃の 1 つとして，暗号が実装されたデバイス等に物理的に干渉を行うことで不正に情報を得るサイドチャネル攻撃があり，タイミング攻撃，電力解析等の様々な種類がある．理論的に安全な方式であっても，サイドチャネル攻撃等によって秘密情報が漏洩する恐れがあるため，NIST の標準化方式/候補に対する実装攻撃対策は重要な研究課題である．実際に UOV に関するサイドチャネル攻撃として，Park らによる電力解析攻撃の報告 [11] がされており，その後も 2023 年,2024 年に Aulbach らによってサイドチャネル攻撃の報告がされている [2][1].

サイドチャネル攻撃に対する対策技術としては，秘密情報をで全て分割した状態で保持・演算を行うマスキングという手法が知られている．安全性モデルとしては，Ishai, Sahai, Wagner によって提案された Probing モデル [9] がよく用いられる．マスキングの安全性評価

* NTT 社会情報研究所, 東京都武蔵野市緑町 3-9-11, NTT Social Informatics Laboratories, 9-11,Midori-Cho 3-Chome Musashino-Shi,Tokyo

は単一の関数で構成される場合は Probing モデルで十分であるが，複数の関数でアルゴリズムが構成される場合は，2016 年に Barthe らが提案した（Strong) Non-Interference ((S)NI) [3] を用いるのが有用である．一方で，多変数多項式暗号に関しては SNI 安全なマスキング方式が提案されていない状況である．

本稿では，初めて QR-UOV に関する SNI 安全なマスキング実装を提案する．QR-UOV では拡大体上の元を主に扱うため，拡大体上の基本演算のマスキング演算を始め，UOV 系の方式で必要な方程式の求解に関するマスキング演算を提案する．また，QR-UOV 全体のマスキング構成案を紹介するとともに，簡単な考察を行う．

2 QR-UOV

本節では，UOV 及び QR-UOV について説明する．

2.1 UOV

q を素数の冪乗とし, $\mathbb{F}_q$ を q 個の要素を持つ有限体とする．さらに, v と m を正の整数とし, $n = v + m$ とする．変数 $\mathbf{x} = (x_1, \ldots, x_n)$ は $\mathbb{F}_q$ 上のベクトルであり, $x_1, \ldots, x_v$ を vinegar 変数, $x_{v+1}, \ldots, x_n$ を oil 変数と呼ぶ.

まず, UOV の鍵生成について説明する．$\mathcal{F} = (f_1, \ldots, f_m) : \mathbb{F}_q^n \to \mathbb{F}_q^m$ を中心写像として構成し, 各 f_k $(k = 1, \ldots, m)$ は次の形の二次多項式であるとする.

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

ここで, $\alpha_{i,j}^{(k)}$ は $\mathbb{F}_q$ の元とする．次に, $\mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ というランダムな線形写像を $\mathcal{F}$ の構造を隠すために選ぶ. 公開鍵 $\mathcal{P}$ は次のような多項式写像として与えられる.

$$\mathcal{P} = \mathcal{F} \circ \mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^m.$$

そして, 秘密鍵は $\mathcal{F}$ と $\mathcal{S}$ で構成される.

次に, 中央写像 $\mathcal{F}$ の逆像の計算方法について説明する．与えられた $\mathbf{y} \in \mathbb{F}_q^m$ に対して, $\mathcal{F}(\mathbf{x}) = \mathbf{y}$ を満たす $\mathbf{x} \in \mathbb{F}_q^n$ を求めるとき, まず, vinegar 変数の値としてランダムに $a_1, \ldots, a_v$ を $\mathbb{F}_q$ から選ぶ. この後, oil 変数 $x_{v+1}, \ldots, x_n$ に関して, 方程式 $\mathcal{F}(a_1, \ldots, a_v, x_{v+1}, \ldots, x_n) = \mathbf{y}$ を解く．これは中心写像の構造 (1) から, m 個の変数, m 個の方程式からなる線形方程式である．もしこの方程式に解がない場合, 新しいランダムな値 $a'_1, \ldots, a'_v$ を選び, 上記の手順を繰り返す.

この逆像の計算方法を使用して, 署名は次のように生成される：署名するメッセージ $\mathbf{m} \in \mathbb{F}_q^m$ が与えられたとき, 方程式 $\mathcal{F}(\mathbf{x}) = \mathbf{m}$ の解 $\mathbf{m}_1$ を求める．これにより, メッセージ $\mathbf{m}$ に対する署名 $\mathbf{s} = \mathcal{S}^{-1}(\mathbf{m}_1) \in \mathbb{F}_q^n$ が得られる．検証の際には, $\mathcal{P}(\mathbf{s}) = \mathbf{m}$ が成立するかどうかを確認する.

UOV の公開鍵および秘密鍵を表す行列について紹介する．公開鍵 $\mathcal{P}$ の各多項式 p_i について, $p_i(\mathbf{x}) = \mathbf{x}^\top \cdot P_i \cdot \mathbf{x}$ となる $n \times n$ 行列 P_i が存在する．同様に, 各 f_i について $n \times n$ 行列 F_i が, $\mathcal{S}$ に対して $\mathcal{S}(\mathbf{x}) = S \cdot \mathbf{x}$ となる $n \times n$ 行列 S が定義される．一般に, これらの行列 P_i および F_i は q が奇数の場合は対称行列として, q が偶数の場合は上三角行列として選ばれる．このとき式 (1) に基づき, q が奇数の場合には F_i は次のような形になる.

$$\begin{pmatrix} *_{v\times v} & *_{v\times m} \\ *_{m\times v} & 0_{m\times m} \end{pmatrix}. \tag{2}$$

さらに, $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$ から,

$$P_i = S^\top F_i S, \ \ (i = 1, \ldots, m),$$

が成り立つ.

2.2 QR-UOV

以下, ℓ を正の整数とし, v と m を ℓ の倍数として選ぶ. このとき, $N := n/\ell$, $V := v/\ell$, $M := m/\ell$ と定義する.

QR-UOV の鍵生成について説明する前に, いくつかの記法を準備する．f を $\mathbb{F}_q[x]$ の元で, 次数 $\deg f = \ell$ であるものとする．剰余環 $\mathbb{F}_q[x]/(f)$ の任意の元 g に対して, $\ell \times \ell$ 行列 Φ_g^f が一意に以下を満たすように定義できる.

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

任意の元 g に対して, 行列 Φ_g^f は ℓ 個の $\mathbb{F}_q$ の要素によって表現できる．$A_f := \left\{ \Phi_g^f \in \mathbb{F}_q^{\ell\times\ell} \,\middle|\, g \in \mathbb{F}_q[x]/(f) \right\}$ は, 行列代数 $\mathbb{F}_q^{\ell\times\ell}$ の部分代数として定義される．このとき, 任意の $X \in A_f$ に対して, WX が対称行列となる可逆行列 $W \in \mathbb{F}_q^{\ell\times\ell}$ が存在することが示されている．特に, f が $x^\ell - ax^i - 1$（$a \neq 0, 1 \leq i < \ell$）の形であるとき, 行列 W は次のようにとることができる.

$$W = \begin{pmatrix} J_i & & \\ & & J_{\ell-i} \end{pmatrix}. \tag{3}$$

ここで, $J_i := \begin{pmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{pmatrix} \in \mathbb{F}_q^{i\times i}$ である．$a, b \in \mathbb{N}$ に対して, $\mathbb{F}_q^{a\ell\times b\ell}$ の部分空間 $A_f^{a,b}$ を次のような元の集合として定義する.

$$\begin{pmatrix} X_{11} & \cdots & X_{1b} \\ \vdots & \ddots & \vdots \\ X_{a1} & \cdots & X_{ab} \end{pmatrix}.$$

ここで, 各 $X_{ij} \in \mathbb{F}_q^{\ell\times\ell}$ は A_f の元である．さらに, $W^{(a)}$ は, $W \in \mathbb{F}_q^{\ell\times\ell}$ を a 個対角に並べた $a\ell \times a\ell$ ブロック対角行列を示す.

これらの記法から, $A_f^{N,N}$ の元と $W^{(N)}A_f^{N,N} := \{A \cdot B \,|\, A \in W^{(N)}, B \in A_f^{N,N}\}$ の元を用いて, QR-UOV の公開鍵と秘密鍵を表現する．ここでは, 公開鍵と秘密鍵を 2.1 節で説明したように行列で表現する．まず, 安全性をの観点から, $f = x^\ell - ax^i - 1 \in \mathbb{F}_q[x]$ で, $a \neq 0$ および $1 \leq i < \ell$ である既約多項式 f と, 式 (3) のような行列 W を準備する．その後, QR-UOV の鍵生成は次のように記述できる：

1. F_i $(i = 1, \ldots, m)$ を, 式 (2) のように右下に $m \times m$ の零行列を持つ対称行列として, $W^{(N)}A_f^{N,N}$ から選ぶ.
2. 可逆な S を $A_f^{N,N}$ からランダムに選ぶ.
3. 公開鍵 $P_i = S^\top F_i S$ $(i = 1, \ldots, m)$ を計算する.

このようにして得られた公開鍵写像 P_i $(i = 1, \ldots, m)$ は, $W^{(N)}A_f^{(N)}$ の元である．署名および検証プロセスは, UOV と同様に行われる．

3 マスキング実装に関する記法と定義

本節では，以降で使用する記法及びマスキングに関する定義を説明する．

3.1 記法

本稿では, ある空間 $\mathbb{K}$ の元 a が演算 $\oplus$ において d 分割されたとき，$d = t + 1$ として, マスキングの分割は $[\![a^{(i)}]\!] = \{a^{(0)}, \ldots, a^{(t)}\}$ と表す.

3.2 マスキング

マスキングでは秘密情報 x を $(x = x_1 \oplus x_2 \oplus \ldots \oplus x_d)$ の d 個に分割を行うことで，サイドチャネル攻撃を用いた秘密情報の漏洩を防ぐことができる．本稿では，特に記載がない場合は和によって分割されているものとする．具体的には t 個のプローブを持つ攻撃者に対して，安全な方式を t-Probing 安全な実装であるといい，2016 年に提案された SNI/NI 安全モデルを利用すると，$d = t + 1$ の分割で t-Probing 安全なマスキングを行うことが可能である.

3.3 (Strong) Non-Interference

以下に (Strong) Non-Interference (SNI/NI) の定義を述べる．[3] で確立された構成定理を用いることで，各ガジェットが t-SNI 安全であることを証明することができればアルゴリズムの安全性を証明することが可能である．SNI の方が NI に比べて強い条件ではあるが，NI であることが証明できれば，後述の t-SNI 安全なリフレッシュ関数を用いることで t-SNI 安全に拡張することができるため，実際には各ガジェットが SNI 安全もしくは NI 安全であることを示せばよい.

Algorithm 1 t-SNI 安全なリフレッシュ関数

Require: $[\![a^{(i)}]\!] = \{a^{(0)}, a^{(1)}, \ldots, a^{(t)}\}$, $(d = t + 1)$
Ensure: $[\![c^{(i)}]\!] = \{c^{(0)}, c^{(1)}, \ldots, c^{(t)}\}$
ただし $\sum_{i=0}^{t} c^{(i)} = \sum_{i=0}^{t} a^{(i)}$

```
for i = 0 to t do
    c^(i) ← a^(i)
end for
for i = 0 to t do
    for j = i + 1 to t do
        r ←$ K
        c^(i) ← c^(i) ⊕ r
        c^(j) ← c^(j) ⊖ r
    end for
end for
```

定義 3.3.1 (t-NI 安全)**.** 分割数 d のガジェット G (入力と出力が d 分割されている) が，任意の $t_1 \leq t$ 個のプローブに対して，t_1 個以下の入力で完全にシミュレーション可能な時，ガジェット G は t-NI 安全であるという.

定義 3.3.2 (t-SNI 安全)**.** 分割数 d のガジェット G(入力と出力が d 分割されている) が，任意の t_1 個の中間値と任意の出力の集合 $O(|O| + t_1 \leq t)$ に対して，t_1 個以下の入力で完全にシミュレーション可能な時，ガジェット G は t-SNI 安全であるという.

3.4 リフレッシュ関数

リフレッシュ関数は，NI 安全なガジェットの出力を入力として実施することで，NI 安全なガジェットを SNI 安全に拡張することができるため，マスキング実装において重要な関数である．Algorithm1 にて，t-SNI 安全なリフレッシュ関数を示す．リフレッシュ関数では，d 個に分割された入力 $[\![a^{(i)}]\!]$ に対して，乱数を用いることで異なる分割 $[\![c^{(i)}]\!]$(ただし,$\sum_{i=1}^{d} a^{(i)} = \sum_{i=1}^{d} c^{(i)}$) に変換を行う.

3.5 Sec-Mult 関数

Sec-Mult 関数は 2016 年に Barthe らが体上の積に関する SNI 安全なマスキング実装として提案した関数 [3] である．Algorithm2 にて，t-SNI 安全な Sec-Mult 関数を示す.

4 QR-UOV のマスキング関数の提案

本節では，QR-UOV のマスキング実装に必要な主要関数を紹介する．QR-UOV の基本演算として必要な和

Algorithm 2 Sec-Mult

Require: $[\![a^{(i)}]\!], [\![b^{(i)}]\!]$
Ensure: $[\![c^{(i)}]\!]$, (ただし $c = a \odot b$)

for $i = 0$ **to** t **do**
  $c^{(i)} \leftarrow a^{(i)} \odot b^{(i)}$
end for
for $i = 0$ **to** t **do**
  for $j = i + 1$ **to** t **do**
    $r \xleftarrow{\$} \mathbb{K}$
    $c^{(i)} \leftarrow c^{(i)} \ominus r$
    $tmp \leftarrow a^{(i)} \cdot b^{(j)}$
    $r \leftarrow r \oplus tmp$
    $tmp \leftarrow a^{(j)} \odot b^{(i)}$
    $r \leftarrow r \oplus tmp$
    $c^{(j)} \leftarrow c^{(j)} \oplus r$
  end for
end for

や転置のマスキングについては，簡単のため本稿では省略する．

4.1 拡大体上の積

拡大体上の元の積を計算するマスキング関数を Algorithm3 に示す．masked_FqL_mul 関数では，入力を拡大体上の元である多項式の係数ベクトル $\mathbf{X}, \mathbf{Y}$ を d 分割した $[\![\mathbf{X}^{(i)}]\!], [\![\mathbf{Y}^{(i)}]\!]$ とし，同じく分割された $[\![\mathbf{Z}^{(i)}]\!]$ を出力する．ただし，出力された $[\![\mathbf{Z}^{(i)}]\!]$ を足し合わせた $\mathbf{Z}$ を係数ベクトルに持つ多項式は，入力である $\mathbf{X}, \mathbf{Y}$ を係数ベクトルに持つ多項式の剰余環 $\mathbb{F}_q[x]/(f)$ 上での積である．多項式の積の演算として，i 次の係数と j 次の係数の積を計算する際には，SecMult 関数を利用している．SecMult 関数は t-SNI 安全である．

4.2 方程式の求解

QR-UOV を始め UOV では，署名生成の際に方程式の求解が必要である（詳細は 2.1 を参照）．本稿では，マスキングにおける安全性を考慮して，マスキング版の逆行列を用いて方程式の求解を行う手法を提案する．

Algorithm4 に方程式の求解を行うマスキング関数を示す．Algorithm4 では，大きく 2 つのステップで方程式の求解を行う．

1 つ目は方程式の係数行列の逆行列を求めるステップで，こちらは Algorithm5 の masked_Mat_inv 関数をサブ関数として呼び出す．masked_Mat_inv 関数は，入力として正則行列が分割された $[\![\mathbf{X}^{(i)}]\!]$ を入力とし，逆行列が分割された $[\![\mathbf{Z}^{(i)}]\!]$ を出力とする関数である．Algorithm5 の step9 終了時点では，分割された $[\![\mathbf{Z}^{(i)}]\!]$ に対して，$\prod \mathbf{Z}^{(i)} = \mathbf{X}^{-1}$ となっている．正当性を示す．

Algorithm 3 masked_FqL_mul：拡大体上の積

Require: $[\![\mathbf{X}^{(i)}]\!], [\![\mathbf{Y}^{(i)}]\!]$（$L$ 個の係数をベクトルとして保持）
Ensure: $[\![\mathbf{Z}^{(i)}]\!]$ (ただし入力とする多項式の積の係数ベクトルの分割)

for $i = 0$ to t **do**
  $\mathbf{T}^{(i)} = \mathbf{0}$
end for
for $i = 0$ to $L - 1$ **do**
  for $j = 0$ to $L - 1$ **do**
    $[\![(Tmp)^{(k)}]\!] \leftarrow \mathbf{SecMult}([\![(X[i])^{(k)}]\!], [\![(Y[j])^{(k)}]\!])$
    for $k = 0$ to t **do**
      $\mathbf{T}^{(k)}[i + j] = \mathbf{T}^{(k)}[i + j] + Tmp^{(k)}$
    end for
  end for
end for
for $i = 2L - 2$ downto L **do**
  for $j = 0$ to t **do**
    $\mathbf{T}^{(j)}[i - L] = \mathbf{T}^{(j)}[i - L] + fc0\mathbf{T}^{(j)}[i]$
    $\mathbf{T}^{(j)}[i - L + fe] = \mathbf{T}^{(j)}[i - L + fe] + fc\mathbf{T}^{(j)}[i]$
  end for
end for
for $i = 0$ to t **do**
  for $j = 0$ to $L - 1$ **do**
    $\mathbf{Z}^{(i)}[j] = \mathbf{T}^{(i)}[j] \pmod q$
  end for
end for

Proof. step9 後での $\mathbf{Z}^0$ は，

$$\begin{aligned}\mathbf{Z}^0 &= \left(\prod_{i=t}^{0} \mathbf{R}_i \mathbf{X}\right)^{-1} \mathbf{R}^t \\ &= \mathbf{X}^{-1} \prod_{i=0}^{t} \mathbf{R}_i^{-1} \mathbf{R}^t\end{aligned}$$

となるため（ここでの $\mathbf{X}$ は入力の値である），

$$\begin{aligned}\prod_{i=0}^{t} \mathbf{Z}^{(i)} &= \mathbf{X}^{-1} \prod_{i=0}^{t} \mathbf{R}_i^{-1} \mathbf{R}^t \prod_{i=t-1}^{0} \mathbf{R}_i \\ &= \mathbf{X}^{-1}\end{aligned}$$

を満たす． □

step10 では，積の分割になっていたものを Mat_M2A 関数で和の分割に変更している．これらの関数で使用している MatRefrsh 関数は，[6] の Appendix に記載されている t-NI 安全な LinearRefresh 関数の行列版である．LinearRefrsh 関数については，元々[12] において提案されている．

Algorithm 4 masked_solve：方程式の求解

Require: $[\![\mathbf{A}^{(i)}]\!] \in \mathbb{F}_q^{m\times m}, [\![\mathbf{y}^{(i)}]\!] \in \mathbb{F}_q^m$
Ensure: $[\![\mathbf{x}^{(i)}]\!]$ (ただし $\mathbf{Ax} = \mathbf{y}$)

for $i = 0$ to t **do**
  $\mathbf{x}^{(i)} = \mathbf{0}$
end for
$[\![\mathbf{A}^{-1(k)}]\!] \leftarrow$ **masked_Mat_inv**$([\![\mathbf{A}^{(k)}]\!])$
for $i = 1$ to m **do**
  for $j = 1$ to m **do**
    $[\![(Tmp)^{(k)}]\!] \leftarrow$ **SecMult**$([\![\mathbf{A}^{-1(k)}[i,j]]\!], [\![\mathbf{y}[j]^{(k)}]\!])$
    for $k = 0$ to t **do**
      $\mathbf{x}[i]^{(k)} \leftarrow \mathbf{x}[i]^{(k)} + Tmp^{(k)}$
    end for
  end for
end for

Algorithm 5 masked_Mat_inv：逆行列

Require: $[\![\mathbf{X}^{(i)}]\!]$, $(\mathbf{X} \in \mathbf{GL}_n(\mathbf{F}))$
Ensure: $[\![\mathbf{Z}^{(i)}]\!]$ (ただし $\mathbf{Z} = \mathbf{X}^{-1}$)

for $i = 0$ to t **do**
  $\mathbf{R}_i \leftarrow \mathbf{GL}_n(\mathbf{F})$
  for $j = 0$ to t **do**
    $\mathbf{X}^{(j)} \leftarrow \mathbf{R}_i \cdot \mathbf{X}^{(j)}$
  end for
  $[\![\mathbf{X}^{(k)}]\!] \leftarrow$ **MatRefresh**$([\![\mathbf{X}^{(k)}]\!])$
  $\mathbf{Z}^{(t-i)} \leftarrow \mathbf{R}_i$
end for
$\mathbf{Z}^{(0)} \leftarrow \mathbf{X}^{-1} \cdot \mathbf{Z}^{(0)}$
$\mathbf{Z} \leftarrow$ **Mat_M2A**$(\mathbf{Z})$

次に逆行列の計算に関して，提案関数の安全性に関して議論する．masked_Mat_inv 関数の安全性に関しては，下記の命題が成り立つ．

命題 4.2.1. Algorithm5(masked_Mat_inv) は t-SNI 安全である．

この命題は下記の 3 つの Lemma から成り立つ．具体的には，Lemma4.2.3, Lemma4.2.4 より，masked_Mat_inv 関数は t-(S)NI な関数で構成されていることが分かる．

Lemma 4.2.2. Algorithm7 の step3 において，MatRefres 関数の入力を除く任意の t 個の値が分かるとき，t 個の値を $t-1$ 個以下の入力で完全にシミュレート可能である．

Proof. 証明については，[5] を参照するものとする．MatRefrsh 関数では，[5] と異なり行列版での構成であるが，証明に関しては同様の手順で行うことができる． □

Algorithm 6 MatRefresh

Require: $[\![\mathbf{X}^{(i)}]\!]$, $(\mathbf{X} \in \mathbf{M}_n(\mathbf{F})$
Ensure: $[\![\mathbf{Z}^{(i)}]\!]$, (ただし $\mathbf{Z} = \mathbf{X}$)

$\mathbf{Z}_t \leftarrow \mathbf{X}_t$
for $i = 0$ to $t-1$ **do**
  $\mathbf{R}_i \leftarrow \mathbf{M}_n(\mathbf{F})$
  $\mathbf{Z}^{(i)} \leftarrow \mathbf{X}^{(i)} + \mathbf{R}_i$
  $\mathbf{Z}^{(t)} \leftarrow \mathbf{Z}^{(t)} - \mathbf{R}_i$
end for

Algorithm 7 Mat_M2A：積から和へのコンバート関数

Require: $[\![\mathbf{X}^{(i)}]\!]$（ただし $\mathbf{X} = \prod_{i=0}^{t}$) $\mathbf{X}^{(i)} \in \mathbf{M}_n(\mathbf{F})$
Ensure: $[\![\mathbf{Z}^{(i)}]\!]$（ただし $\mathbf{Z} = \sum_{i=0}^{t} \mathbf{Z}^{(i)} = \mathbf{X}$)

$\mathbf{Z}^{(0)} \leftarrow \mathbf{X}^{(0)}$
for $i = 0$ to $t-1$ **do**
  $\mathbf{Z}^{(0)}, \ldots, \mathbf{Z}^{(i+1)} \leftarrow$ **MatRefresh**$(\mathbf{Z}^{(0)}, \ldots, \mathbf{Z}^{(i)}, \mathbf{0})$
  for $j = 0$ to $i+1$ **do**
    $\mathbf{Z}^{(j)} \leftarrow \mathbf{Z}^{(j)} \cdot \mathbf{X}^{(i+1)}$
  end for
end for

Lemma 4.2.3. Algorithm7(積から和へのコンバート関数) は t-NI 安全である．

Proof. Algorithm7 に対して，入力を $\mathbf{X}^{(0)}, \ldots, \mathbf{X}^{(i)}$ とし，出力を $i-1$ 番目の for 文終了時，すなわち $\mathbf{Z}^{(0)}, \ldots, \mathbf{Z}^{(i)}$ とするガジェットを G_i とおく．ただし $\mathbf{Z}^{(0)} \leftarrow \mathbf{X}^{(0)}$ をガジェット G_0 とする．このように定義すると，Algorithm7 はガジェット G_t である．また各 for 文 i の実行におけるガジェットを H_i とおく．すなわち，入力は $\mathbf{Z}^{(0)}, \ldots, \mathbf{Z}^{(i)}, \mathbf{X}^{(i+1)}$ とし，出力は $\mathbf{Z}^{(0)}, \ldots, \mathbf{Z}^{(i+1)}$ である．定義の仕方より，ガジェット G_i はガジェット G_{i-1} と H_{i-1} との合成である．

$G_i (0 \le i \le t)$ において t-NI 安全であることを示せば，Algorithm7 は t-NI であることを示せる．上記を帰納法で示す．G_0 においては自明である．次に G_{i-1} が任意の $i-1$ 個の中間値に対して，$i-1$ 個以下の入力でシミュレート可能であると仮定して，ガジェット G_i について考える．ガジェット G_i において，i 個の中間値の集合を M をとってくる．この時 M の元は，G_{i-1} または H_{i-1} のどちらかのみに属する．すなわち，それぞれに属する中間値の個数を t_1, t_2 とすると，$t_1 + t_2 = i$ である．

$t_2 = 0$ の時，すなわち M の元は全て G_{i-1} に属する時は，仮定より $t_1 \le i$ 個以下の入力でシミュレート可能である．

$t_2 \neq 0$ である時，すなわち M の元のうち少なくとも 1 つは H_{i-1} の元に属する時は，t_2 個の中間値がさらに MatRefresh 及びその出力に属する場合とその他で分け

ることができる．t_2 個の中間値において，H_{i-1} の際の step3 に際して Lemma4.2.2 を適用すると，t_2-1 個の入力でシミュレート可能である．ただし，その他の場合では $\mathbf{X}^{(i)}$ を用いることでシミュレート可能であるため，i をシミュレートするための入力に加える必要がある．すなわち，ガジェット G_i を完全にシミュレートするための入力 I は，$I_i = I_{i-1} \cup I_{t2} \cup \{i\}$ であり，$\#I_i \leq t_1 + t_2 - 1 + 1 = i$ である．よって i に対して成り立つことが示せたため，帰納的にガジェット G_t，すなわち Mat_M2A 関数は t-NI 安全であることが示せた． □

Lemma 4.2.4. Algorithm5(masked_Mat_inv) は，step10 でのコンバート関数を除いたガジェットとした時，t-SNI 安全である．

Proof. 任意の t_1 個の中間値の集合 M と任意の出力の集合 $O(|O| + t_1 \leq t)$ に対して，(M, O) を完全にシミュレーション可能な入力の部分集合 $I \coloneqq \{\mathbf{a}_i \mid i \in \mathcal{I}\}$ を下記の手順で構成する．その時に $\#I \leq t_1$ であることを示す．step1 から step8 での for 文の間を添え字 i を用いて Part_i とし，Part_i 終了時の $\mathbf{X}^{(j)}$ の値を $\mathbf{X}_i^{(j)}$ とする．ここで，I は初めは空集合とし，もし $\mathbf{X}^{(j)} \in M$ であれば j を I に加える．この時，$\#I \leq \#M \leq t_1$ である．

また，$\#M + \#O \leq t < t+1$ であるため，Part_i や $\mathcal{O}$ に属していない最小の $i^* \subset [0, t]$ が存在する．i^* を基準として場合分けを行う．中間値 M に含まれる元が，Part_i の元，ただし $i < i^*$ である時，I にてシミュレート可能である．$i > i^*$ に関しては，i^* の取り方から，$\mathbf{R}_{i^*}, \mathbf{Z}_{i^*}$ は，中間値及び出力の集合に含まれていないため，$\mathbf{X}_{i^*}^j$ は完全なランダムな行列である．それ以降の i についても同様である．よってシミュレート可能である．$\mathbf{R}_i$ 及び出力の集合については，n 次元の正則行列全体から一様ランダムに取ってきたものなので、シミュレート可能である．よって t-SNI 安全である． □

2 つ目のステップは，逆行列を方程式の両辺に掛けることで，解を求めるステップであり，こちらは行列とベクトルの各要素の積に Sec_Mult 関数を利用することで計算可能である．

5 QR-UOV のマスキング実装の提案

本節では，QR-UOV 全体 (鍵生成/署名生成) のマスキング実装を提案するとともに，一部マスキング関数の性能報告を行う．鍵生成に関しては Algorihtm8 で，署名生成に関しては Algorithm9 にて示す．ただし，Algorithm9 内の関数 ϕ は，行列の各成分 $g \in \mathbb{F}_{q^\ell}$ を Φ_g^f へと移す関数である．QR-UOV では，秘密情報が関わる演算として，和/積のような基本演算や，各秘密情報のサンプル，

Algorithm 8 マスキング版：KeyGen()

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

1: $\mathsf{seed}_{\mathsf{pk}}, [\![\mathsf{seed}_{\mathsf{sk}}^{(i)}]\!] \xleftarrow{\$} \{0,1\}^{2\lambda}$
2: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$
▷ $P_{i,1} \in \mathbb{F}_{q^\ell}^{V\times V}$
▷ $P_{i,2} \in \mathbb{F}_{q^\ell}^{V\times M}$
3: $S'^{(0)}, S'^{(1)}, \ldots, S'^{(t)} \leftarrow \mathsf{Expand}_{\mathsf{sk}}([\![\mathsf{seed}_{\mathsf{sk}}^{(i)}]\!])$
▷ $S'^{(i)} \in \mathbb{F}_{q^\ell}^{V\times M}$
4: **for** i from 1 to m **do**
5: $P_{i,3} \leftarrow -[\![S'^{\top(i)}]\!]P_{i,1}[\![S'^{(i)}]\!] + P_{i,2}^{\top}[\![S'^{(i)}]\!] + [\![S'^{\top(i)}]\!]P_{i,2}$
6: **end for**
7: **return** $(\mathsf{pk}, \mathsf{sk}) = \left(\left(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}\right), [\![\mathsf{seed}_{\mathsf{sk}}^{(i)}]\!]\right)$

方程式の求解などがあり，それら全てを分割した状態で行う必要がある．

5.1 基本演算

基本演算に関しては，各成分を拡大体上の元とする行列の和/積及び転置によって構成されている．それぞれ masked_Mat_add/masked_Mat_mul/masked_FqL_tra 関数とする．例えば鍵生成時（Algorithm8 の step5）では，まず $[\![S'^{\top(i)}]\!] \leftarrow$ masked_Mat_tra($[\![S'^{(i)}]\!]$) で秘密情報 $[\![S'^{(i)}]\!]$ の転置計算を行う．次に公開情報と秘密情報の積を halfmasked_Mat_mul で計算する．halfmasked 関数は，2 つの入力のうち片方が公開情報，すなわち分割されていない状態である際に用いる関数である．step5 では，$[\![S'^{\top(i)}]\!]P_{i,1}[\![S'^{(i)}]\!]$, $P_{i,2}^{\top}[\![S'^{(i)}]\!]$, $[\![S'^{\top(i)}]\!]P_{i,2}$ の計算を行う．最後にこれらを値の和を取る必要があり，それには masked_Mat_add で計算を行う．このように QR-UOV では masked_Mat_add/masked_Mat_mul/masked_FqL_tra 関数やその halfmasked 関数を利用することで基本的な演算を行うことができる．鍵生成アルゴリズム (Algorithm8)，署名生成アルゴリズム（Algorithm9）の疑似コードでは，これらの基本演算に関しては省略して記載している．

4 章と同様で，本稿では積（masked_Mat_mul）に関してのみ述べる．行列の積の各演算は，行列の成分毎の積及びその積の値同士の和の 2 つの基本演算で完結する．この基本演算にサブ関数として，拡大体上の積のマスキング関数 (masked_Fql_mul) を利用することで，QR-UOV 内の行列の積に関しては t-SNI 安全なマスキング関数を利用することができる．本提案とは別に，乱数を行列単位で取得することで，行列単位の積として直接マスキング関数を構成することも可能である．

Algorithm 9 マスキング版：Sign($\mathbf{M}, \mathsf{pk}, \mathsf{sk}$)

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

Output: signature σ

$\left(\mathsf{seed}_{\mathsf{pk}}, \{P_{i,3}\}_{i\in[m]}\right) \leftarrow \mathsf{pk}$

$[\![\mathsf{seed}_{\mathsf{sk}}^{(i)}]\!] \leftarrow \mathsf{sk}$

$\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathsf{Expand}_{\mathsf{pk}}(\mathsf{seed}_{\mathsf{pk}})$

$S'^{(0)}, S'^{(1)}, \ldots, S'^{(t)} \leftarrow \mathsf{Expand}_{\mathsf{sk}}([\![\mathsf{seed}_{\mathsf{sk}}^{(i)}]\!])$

for i from 1 to m **do**

  $F_{i,1} \leftarrow \phi P_{i,1}$

  $[\![F_{i,2}^{(k)}]\!] \leftarrow \phi(-P_{i,1}[\![S'^{(k)}]\!] + P_{i,2})$

end for

$r_0, r_1, \ldots, r_t \xleftarrow{\$} \{0,1\}^{\lambda}$

$[\![\mathbf{t}^{(k)}]\!] \leftarrow \mathsf{Hash}([\![\mathbf{M}^{(k)}]\!] | [\![r^{(k)}]\!])$ ▷ $\mathbf{t} \in \mathbb{F}_q^m$

repeat

  for i from 0 to t **do**

    $\mathbf{y}^{(i)} = (y_1^{(i)}, \ldots, y_v^{(i)})^{\top} \xleftarrow{\$} \mathbb{F}_q^v$

  end for

  $[\![L^{(k)}]\!] \leftarrow \begin{pmatrix} 2[\![\mathbf{y}^{\top(k)}]\!][\![F_{1,2}^{(k)}]\!] \\ \vdots \\ 2[\![\mathbf{y}^{\top(k)}]\!][\![F_{m,2}^{(k)}]\!] \end{pmatrix}$ ▷ $L \in \mathbb{F}_q^{m\times m}$

until L is full rank

$[\![\mathbf{u}^{(k)}]\!] \leftarrow \left([\![\mathbf{y}^{\top(k)}]\!]F_{1,1}[\![\mathbf{y}^{(k)}]\!], \ldots, [\![\mathbf{y}^{\top(k)}]\!]F_{m,1}[\![\mathbf{y}^{(k)}]\!]\right)^{\top}$ ▷ $\mathbf{u}^{(k)} \in \mathbb{F}_q^m$

$[\![\mathbf{x}^{(k)}]\!] \leftarrow \mathsf{masked\_solve}([\![L^{(k)}]\!], [\![\mathbf{t}^{(k)}]\!] - [\![\mathbf{u}^{(k)}]\!])$

$[\![\mathbf{s}^{(k)}]\!] \leftarrow \begin{pmatrix} [\![\mathbf{y}^{\top(k)}]\!] - [\![S'^{(k)}]\!][\![\mathbf{x}^{\top(k)}]\!] \\ [\![\mathbf{x}^{\top(k)}]\!] \end{pmatrix}$

return $\sigma = (r, \mathbf{s})$

5.2 秘密情報のサンプル

QR-UOV の秘密情報のサンプルとして鍵生成時の seed の生成，秘密鍵に関する S' の生成，署名生成時の $\mathbf{y}, r$ の生成がある．これらは，一様ランダムにサンプルすれば良いので，本提案のマスキング実装では，マスキングオーダである d 回ほど通常のサンプルを行い，それぞれを分割されたものとみなしている．すなわち，d 回サンプルされたものを足し合わせてたものが，マスキングしない通常のサンプルである．効率性に関しては検討していないため，サンプルに関してはまだ検討の余地がある．

5.3 方程式の求解

第 1 ラウンドでの QR-UOV 仕様では，方程式の求解部分に関して LU 分解を利用している．我々が提案するマスキングでは，行列 L が正則であると仮定して，行列 L の逆行列を用いることで方程式の求解を行う．もし正則でない場合を考慮するのであれば，step11 に戻り行列 L を取り直すことで可能である．分割されている行列を足し合わせた時に正則であるかを判定するには，masked_Mat_inv 関数を用いればよい．具体的には，Algorithm5 の step9 の際に，逆行列 $\mathbf{X}^{(-1)}$ が存在するかで判定することができる．そのためには，Algorithm5 の乱数として正則行列を使用する必要がある．署名アルゴリズムとしては，行列 L を生成した後の step16 で，masked_Mat_inv 関数で正則性の判定及び正則なら逆行列を出力させ，正則でないなら step11 に戻り行列 L を取り直し，正則であるならば逆行列 L^{-1} を masked_solve の入力に付け加えて方程式の求解を実施すればよい．この場合は，masled_solve 関数内では，逆行列を求める計算 (masked_Mat_inv) を行わない．

表 1: masked_Fql_mul の平均実行時間 (μ 秒)

マスキングオーダ	0	1	2	3
qruov1q31L3	0.01	0.98	1.38	4.11
qruov3q31L3	0.01	0.99	1.39	1.89
qruov5q31L3	0.01	0.98	1.53	1.69

表 2: masked_Mat_inv の平均実行時間 (μ 秒)

マスキングオーダ	0	1	2	3
qruov1q31L3	1299	6641	11113	21158
qruov3q31L3	4601	22376	36396	65205
qruov5q31L3	8092	39165	87801	160948

5.4 マスキング関数の性能について

masked_Fql_mul 及び masked_Mat_inv に関しての性能測定を実施した．実行環境は，Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz で, OS は Ubuntu24.04 LTS である．masked_Fql_mul 及び masked_Mat_inv に関しての性能結果をそれぞれ表 1，表 2 にて示す.

各表は，NIST に提案されている QR-UOV のセキュリティパラメータ 1,3,5 の $(q = 31, \ell = 3)$ において，マスキング無し，マスキング有（マスキングオーダ d は 1 から 3 まで）における各関数の実行時間 (μ 秒) を示している．ただし，実行時間は各 100 回における平均時間を表しており，表のマスキングオーダが 0 はマスキングなしを意味している．また，逆行列の測定に関しては，あらかじめ正則だと判定されたものだけを入力として測定を実施している．

拡大体上の積の計算においては，マスキングの有無によって大きく実行時間が増加しているが，マスキングオーダの増加による影響は緩やかである．元の関数自体の実行時間は短いため単体の関数としてはそれほど時間はかからないが，積の演算の回数は多いため，全体に与える影響については検討が必要である．

逆行列の計算に関しては，積のマスキングに比べて，マスキングの有無における実行時間の増加割合は少なく，

マスキングオーダの増加による実行時間の増加割合は少し多い．関数の実行時間は積の演算に比べて長いが，実行回数はたかだか数回であるため，こちらも全体に与える影響については検討が必要である．

6 まとめと今後の展望

本稿では，QR-UOV のマスキング実装を初めて提案した．t-SNI 安全なモデルでの構成のため，方程式の求解には逆行列を用いており，逆行列を求めるマスキング関数等も提案した．一方でサンプル部分に関しては，シンプルにマスキングオーダ数のサンプル生成を行うなど，効率面での検討はまだ不十分である．今後は，QR-UOV 全体のマスキング実装を実機で測定するとともに，効率面での評価及び改良を検討していく．

参考文献

[1] Thomas Aulbach, Fabio Campos, and Juliane Krämer. Sok: On the physical security of uov-based signature schemes. *Cryptology ePrint Archive*, 2024.

[2] Thomas Aulbach, Fabio Campos, Juliane Krämer, Simona Samardjiska, and Marc Stöttinger. Separating oil and vinegar with a single trace: side-channel assisted kipnis-shamir attack on uov. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):221–245, 2023.

[3] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 116–129, 2016.

[4] Ward Beullens. Mayo: practical post-quantum signatures from oil-and-vinegar maps. In *International Conference on Selected Areas in Cryptography*, pages 355–376. Springer, 2021.

[5] Jean-Sébastien Coron. High-order conversion from boolean to arithmetic masking. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 93–114. Springer, 2017.

[6] Jean-Sébastien Coron, François Gérard, Matthias Trannoy, and Rina Zeitoun. High-order masking of ntru. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 180–211, 2023.

[7] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. A new variant of unbalanced oil and vinegar using quotient ring: Qr-uov. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part IV 27*, pages 187–217. Springer, 2021.

[8] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[9] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 463–481. Springer, 2003.

[10] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 206–222. Springer, 1999.

[11] Aesun Park, Kyung-Ah Shim, Namhun Koo, and Dong-Guk Han. Side-channel attacks on post-quantum signature schemes based on multivariate quadratic equations:-rainbow and uov. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 500–523, 2018.

[12] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 413–427. Springer, 2010.

[13] The National Institute of Standards and Technology (NIST). Post-quantum cryptography. `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization`.

[14] C.-Y. Ding J. Kuan Y.-L. Li M.-S. Tseng B.-S. Tseng P.-E. Wang C.-C. Wang L.-C., Chou. Snova: Proposal for nistpqc: Digital signature schemes project. `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/SNOVA-spec-web.pdf`.