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 の定数時間実装の提案
Constant-Time Implementation of QR-UOV

秋山 梨佳 * 金城 皓羽 * ティブシ メディ *
Rika Akiyama Koha Kinjo Mehdi Tibouchi

あらまし デジタル署名方式 QR-UOV は NIST の耐量子計算機暗号標準化コンペティションに提出されており, 2024 年 10 月にコンペティション第 2 ラウンドへ進出したことが発表されている. 本稿では, QR-UOV のタイミング攻撃対策として, 定数時間実装の提案と実装評価を行う. QR-UOV の署名アルゴリズムでの, タイミングリークが発生する箇所の一つに, 連立方程式求解部がある. 本稿で提案する定数時間実装では, 連立方程式求解部に定数時間ガウスの消去法を組み込むことで, 定数時間処理を実現した. 実装評価では, dudect と TIMECOP を用いてタイミングリークのリスクと性能を測定した. その結果, 提案方式について, 現仕様で検出されていたタイミングリーク箇所をすべてクリアできたこと, また処理速度は 1.1 倍の範囲内に抑えられていることを確認した.

1 導入

RSA 暗号や楕円曲線暗号など現在広く使われている公開鍵暗号は, 量子コンピュータ上で実行可能なショアのアルゴリズム [15] により, 多項式時間で解読されることが知られている. 大規模な量子計算機の出現により, 現行公開鍵暗号が危殆化してしまうことから, 量子計算機の解読に耐性のある耐量子計算機暗号 (Post-Quantum Cryptography(PQC)) の研究開発が盛んになっている. 米国国立標準技術研究所 (NIST) では, 2016 年から PQC 標準化に向けた公募を開始し, その過程で新たに, 2022 年 6 月から署名方式の追加公募を開始している [12].

デジタル署名方式 QR-UOV[5] は ASIACRYPT 2021 で古江らによって提案された方式で, 既存方式である UOV をベースとして鍵の行列表現に剰余環を導入し, 公開鍵長の削減を実現している. NIST の署名追加公募にも提出されており [4], 2024 年 10 月に発表された Round2 への進出方式の一つとして選ばれている. QR-UOV のリファレンスコードは公開されており [13], 星野らによる効率的な実装手法 [7] が取り入れられている.

一方, NIST は公募要件で, 定数時間実装は最低限用意すべきと言及しており [12], QR-UOV の実装においても定数時間実装の評価は重要な観点である. 本稿では, QR-UOV の定数時間実装の提案をする. はじめに, QR-UOV の現仕様版（リファレンスコード）を二つの観点 (実行時間のばらつきと具体的なタイミングリーク箇所の検出) から測定を行い, 定数時間処理を施すべき部分をまず明らかにした. 次に, 検出されたタイミングリーク箇所に対し, 攻撃の脅威性の有無を判断し, 脅威性有りと判断した箇所にそれぞれ定数時間処理を施すことで, QR-UOV の定数時間実装を構成した. また, 提案方式についてタイミングリークが抑えられていることを確認し, 処理速度の変化についても測定をした.

本稿の構成は次の通りである. 2 節では, QR-UOV とタイミング攻撃の起点について説明する. 3 節では, タイミングリークの一般的な測定方法について説明し, 実際に QR-UOV の現仕様版を測定した結果を示す. 4 節では, 現仕様版の測定結果を踏まえた上での対策版（QR-UOV の定数時間実装）を提示し, こちらについても測定を行い, 得られた結果から考察をする.

2 準備

2.1 QR-UOV 署名方式

耐量子計算機暗号の一つに, 多変数多項式暗号 (Multivariate Public Key Cryptography(MPKC)) がある. MPKC は, 有限体上の二次多項式系の求解問題 (MQ 問題) に基づいて構成される公開鍵暗号である. $\mathbb{F}_q$ を素数 q の有限体とする. n を変数の数, m を多項式の数としたとき, 次のような, 有限体 $\mathbb{F}_q$ 上の連立二次方程式の求解問題を考える: $\mathbb{F}_q$ 上の n 変数 2 次多項式系

* NTT 社会情報研究所, 〒 180-8585 東京都武蔵野市緑町 3-9-11, NTT Social Informatics Laboratories, 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan. ({rika.akiyama, kouha.kinjo, mehdi.tibouchi} @ntt.com)

$\mathcal{P} = (p_1, \cdots, p_m) \in \mathbb{F}_q[x_1, \cdots, x_n]^m$ が与えられたとき, $\mathcal{P}(a_1, \cdots, a_n) = 0 \in \mathbb{F}_q^m$ を満たす解 $(a_1, \cdots, a_n) \in \mathbb{F}_q^n$ を求める. このような問題を MQ 問題と呼ぶ. MQ 問題は NP 完全であることが証明されており [6], MPKC の安全性の根拠となっている. QR-UOV は MPKC ベースの電子署名方式であり, 電子署名方式とは, 以下を満たすアルゴリズムの組 (KeyGen, Sign, Verify) である：

1. 鍵生成 $\mathsf{KeyGen}(1^\kappa) \to (\mathsf{sk}, \mathsf{pk})$

   セキュリティパラメータ κ に基づき, 署名鍵 sk と検証鍵 pk を出力する.

2. 署名生成 $\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$

   署名鍵 sk を用いてメッセージ M の署名 σ を出力する.

3. 検証 $\mathsf{Verify}(M, \sigma, \mathsf{pk}) \to$accept または reject

   検証が成功した場合に accept を, そうでない場合は reject を出力する.

以下, $v, o \in \mathbb{Z}_{>0}$ とし, $n := v + o$ とする. MPKC の署名方式は次のように定義される:

1. 鍵生成 $\mathsf{KeyGen}(1^\kappa) \to (\mathsf{sk}, \mathsf{pk})$

   セキュリティパラメータ κ に基づき, 署名鍵 $\mathsf{sk} = \mathcal{F} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ と検証鍵 $\mathsf{pk} = \mathcal{P} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ を出力する.

2. 署名生成 $\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$

   署名鍵 sk を用いてメッセージ $M \in \mathbb{F}_q^n$ の署名 $\sigma \in \mathbb{F}_q^n$ を出力する.

3. 検証 $\mathsf{Verify}(M, \sigma, \mathsf{pk}) \to$accept または reject

   $\mathcal{P}(\sigma) = M$ ならば accept を, そうでないならば reject を出力する.

MPKC の署名方式の一つである UOV 署名 [8] について説明する. $\mathbf{x}_v := (x_1, \cdots, x_v), \mathbf{x}_o := (x_{v+1}, \cdots, x_n), \mathbf{x} = (\mathbf{x}_v, \mathbf{x}_o)$ とおき, $\mathbf{x}_v$ を vinegar 変数, $\mathbf{x}_o$ を oil 変数と呼ぶ. このとき UOV 署名方式は, 以下を満たすアルゴリズムの組 (KeyGen, Sign, Verify) として定義される:

1. 鍵生成 $\mathsf{KeyGen}(1^\kappa) \to (\mathsf{sk}, \mathsf{pk})$

   可逆な写像 $\mathcal{S} : \mathbb{F}_q^n \to \mathbb{F}_q^n$ をランダムに取る. また, $\mathcal{F} = (f_1, \cdots, f_m) : \mathbb{F}_q^n \to \mathbb{F}_q^m$ をとる. ただし各 f_k $(k = 1, \cdots, m)$ は次の形をしているものとする:

   $$f_k(\mathbf{x}) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{i,j}^{(k)} x_i x_j, \quad (\alpha_{i,j}^{(k)} \in \mathbb{F}_q).$$

   このとき, 署名鍵 sk を $(\mathcal{F}, \mathcal{S})$, 検証鍵 pk を $\mathcal{P} = \mathcal{F} \circ \mathcal{S}$ とする.

2. 署名生成 $\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$

   署名鍵 sk を用いてメッセージ $M \in \mathbb{F}_q^n$ の署名 $\sigma := \mathcal{S}^{-1}(x) \in \mathbb{F}_q^n$ を出力する. ただしここで x は $\mathcal{F}(x) = M$ の解としている. 解 x の求め方は次の通り:

   vinegar 変数 $\mathbf{x}_v$ をランダムに取り $\mathcal{F}$ に代入すると, $\mathcal{F}$ は変数 $x_{v+1}, \cdots x_n$ に関する連立一次方程式となる. これを解き, 解が存在すればその解を x とし, 解が存在しなければ vinegar 変数を取り直すことに戻る. これを解 x が得られるまで繰り返す.

3. 検証 $\mathsf{Verify}(M, \sigma, \mathsf{pk}) \to$accept または reject

   $\mathcal{P}(\sigma) = M$ ならば accept を, そうでないならば reject を出力する.

UOV 署名は, 署名長が短く検証速度が速いという利点を有する一方で, 公開鍵長が長いことが欠点とされている.

QR-UOV 署名は, UOV 署名の公開鍵と秘密鍵に剰余多項式環の構造を導入することで構成される. これにより, UOV 署名の利点は保持し, 欠点であった公開鍵長を削減した方式となっている. ここでは, QR-UOV で導入した追加構造の部分に絞って説明をする. 詳細は [5] を参照されたい. l を正の整数とし, f を次数 l の $\mathbb{F}_q[x]$ の多項式とする. このとき, 剰余多項式環 $\mathbb{F}_q[x]/(f)$ の任意の元 g に対し, 次を満たすような $\mathbb{F}_q$ 上の $l \times l$ 行列 Φ_g^f が一意に定まる:

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

QR-UOV 署名は, vinegar 変数と oil 変数の数をそれぞれ l の倍数とし, 鍵構造に行列 Φ_g^f を導入することで, $l \times l$ サイズの行列が l 個の要素で表現可能となり, 鍵サイズをおおよそ l 分の 1 に削減している.

本稿では, QR-UOV 署名方式のアルゴリズム内で秘密情報を扱う部分である, 署名生成アルゴリズム (Algorithm1) に対し定数時間処理を施すことを考える.

2.2 タイミング攻撃の起点と定数時間実装

暗号アルゴリズムを実装した暗号モジュールに対する攻撃は実装攻撃と呼ばれ, 近年現実的な脅威として認識され始めている. 暗号モジュールの開発では従来, 高速化や低コスト化を目指した設計に主眼が置かれていたが, 実装攻撃の台頭に伴い, 実装攻撃に対する安全性の研究も重要な課題の一つとなっている.

暗号モジュールに対する実装攻撃は, 破壊型攻撃と非破壊型攻撃（サイドチャネル攻撃）に大別される. 前者は比較的古くから知られている攻撃手法であり, 一般に高度な物理的解析や高価な装置を伴うため, 実行できる攻撃者は限定されることが多い. 一方後者は 1990 年代

Algorithm 1 Sign($\mathbf{M}$, $\mathtt{pk}$, $\mathtt{sk}$)

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

Output: signature σ

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

後半から提案され始めた [9] 手法であり, 比較的安価な装置で実現できるため, 現実的な脅威となっている.

サイドチャネル攻撃とは, 暗号モジュールから漏れる物理的な特性を外部から測定することでモジュール内の秘密の情報を推定する攻撃であり, 着目する物理的特性（処理時間, 消費電力・電磁波, 音など）により様々攻撃の種類がある. その中でも特に, 処理時間の差異に着目したタイミング攻撃への対策に当たる定数時間実装は, 実装技術の中でも基礎的なものであり, 各種攻撃とその対策の中でも優先的に検討すべきものに当たる.

定数時間実装の検討の際, 一般的に着目すべきタイミングリークの基本的な要因は下記の 3 点である：

1. 秘密情報に依存する条件分岐
   (e.g. if ⟨ secret == 0 ⟩ then {⋯} else {⋯}),
2. 秘密情報に依存するテーブル参照
   (e.g. s[i] = substitution_table[secret[i]]),
3. 秘密情報により実行時間が変化する CPU 操作
   (e.g. x = ⟨ secret ⟩ / y).

これらの要因によりタイミングリークが起きている箇所に対し, 適切に定数時間処理を施すことで定数時間実装を構成する.

3 課題の抽出

3.1 タイミングリークの測定方法について

タイミングリークの測定方法として, dudect[14] と TIMECOP[10] を用いる.

dudect とは, 統計的手法（Welch の t 検定）を用いて実行時間のばらつきを検出するツールである. 2 つの異なる入力データクラスについて実行時間を測定し, その 2 つのタイミング分布が統計的に異なるか否かを, t 値を計算することで判定する. 一般的には, t 値が 5 以下であれば大きなタイミング漏れがないと判断できる.

TIMECOP は, タイミングリークの起きている箇所を具体的に検出するツールである. より詳細には, Valgrind[11] の memcheck 機能を用いることで, タイミングリーク懸念箇所を検出している. Valgrind はソフトウェアのテスト技術における, 動的解析ツールの一つである. TIMECOP の出力結果は三種類ある. 実行アルゴリズム全体を通して TIMECOP がタイミングリークを検出しなかった場合は timecop_pass と検出され, リークを検出した場合は timecop_fail と出力される. また, 実装上にバグが存在していたり, Valgrind が失敗した場合は timecop_error と出力される. timecop_fail は誤検出する (定数時間な実装であるが fail と出力する) 場合もある. そのため, TIMECOP を用いた定数時間実装の検討の際は, 測定結果と実装を再度照らし合わせ, fail として出力された箇所の脅威性の有無を判断する作業が必要になる. TIMECOP は 2022 年に SUPERCOP[1] に集約され, 測定したい実装を SUPERCOP にかけることで TIMECOP の結果も得られるようになっている.

本研究では, これら二つのツールを用いることで相補的な評価を行い, より精度の高い定数時間実装を目指した.

3.2 現仕様版の測定

本稿での実験は, Intel Core i9-9880H(2.30GHz), Virtualbox 7.0.4, Ubuntu 24.04 を使用し, vCPU 4 コア, メモリ 16GB の環境で実施. また, 先述した二つのツール (dudect と SUPERCOP) を用いて QR-UOV の現仕様版の測定を行う.

まず, dudect による測定結果を図 1 に示す. 横軸が実行回数, 縦軸が t 値となっている. 測定の結果, $1M = 10^6$ 回までの測定では t 値が 5 を超えることはなく, おおむね 2 から 2.5 の幅で安定する結果となった. これより, QR-UOV は現仕様版の時点で既に, 実行時間に大きなばらつきはないことがわかる.

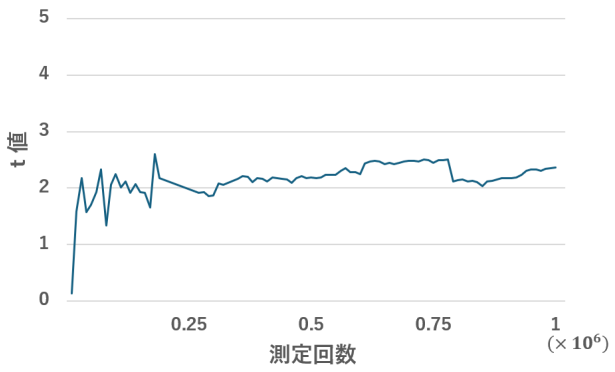

図 1: dudect による現仕様版のタイミングリーク評価

次に, TIMECOP による測定結果を示す. 現仕様版を測定した結果, 具体的なタイミングリークの懸念箇所 (timecop_fail) として下記が検出された：

1. 乱数サンプリングを行う棄却サンプリング,
2. 連立方程式求解アルゴリズムの一部である LU 分解,
3. 逆元のテーブル参照.

棄却サンプリングについて, TIMECOP での検出結果に基づいて実際に実装を確認したところ, q か q 以外の数字かで分岐する実装箇所があり, q に関する情報が洩れる可能性がある. しかし,

- 今回 q は公開情報であること
- 疑似乱数生成器の疑似乱数性を仮定すれば, seed に関する情報が洩れないこと

のこれら二つの理由により, 脅威性はないと結論付けた. 一方で, LU 分解ではタイミングリークの秘密情報による条件分岐が, 逆元のテーブル参照では秘密情報によるテーブル参照が起きているため, これら二つに対し定数時間処理を施すことを検討した.

4 提案方式

4.1 QR-UOV の定数時間化の提案

現仕様をベースに, QR-UOV の定数時間実装を検討した (Algorithm2). 現仕様版と比較して下記 2 か所を変更した：

1. 定数時間ガウスの消去法の導入
2. 逆元のテーブル参照部分の書き換え

変更点 1 について, 現仕様では連立方程式求解部は LU 分解を用いた手法となっており, ここがタイミングリーク箇所として検出されている (Alg1. L15). そこで提案

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

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

1: $(\mathtt{seed}_{\mathtt{pk}}, \{P_{i,3}\}_{i\in[m]}) \leftarrow \mathtt{pk}$
2: $\mathtt{seed}_{\mathtt{sk}} \leftarrow \mathtt{sk}$
3: $\{P_{i,1}\}_{i\in[m]}, \{P_{i,2}\}_{i\in[m]} \leftarrow \mathtt{Expand}_{\mathtt{pk}}(\mathtt{seed}_{\mathtt{pk}})$
4: $S' \leftarrow \mathtt{Expand}_{\mathtt{sk}}(\mathtt{seed}_{\mathtt{sk}})$
5: **for** i from 1 to m **do**
6: $\quad F_{i,1} \leftarrow P_{i,1}$
7: $\quad F_{i,2} \leftarrow -P_{i,1}S' + P_{i,2}$
8: $S \leftarrow \begin{pmatrix} I_v & S' \\ 0_{m\times v} & I_m \end{pmatrix}$
9: $r \xleftarrow{\$} \{0,1\}^{\lambda}$
10: $\mathbf{t} \leftarrow \mathtt{Hash}(\mathbf{M}||r)$
11: $\mathbf{y} = (y_1, \cdots, y_v)^{\top} \xleftarrow{\$} \mathbb{F}_q^v$
12: $L \leftarrow \begin{pmatrix} 2\mathbf{y}^{\top}F_{1,2} \\ \vdots \\ 2\mathbf{y}^{\top}F_{m,2} \end{pmatrix}$
13: $\mathbf{u} \leftarrow \left(\mathbf{y}^{\top}F_{1,1}\mathbf{y}, \cdots, \mathbf{y}^{\top}F_{m,1}\mathbf{y}\right)^{\top}$
14: $\mathbf{x} = \mathtt{Elim}(L, \mathbf{t}-\mathbf{u})$
15: $\mathbf{s} \leftarrow S^{-1}\left(y_1, \cdots, y_v, x_1, \cdots, x_m\right)^{\top}$
16: **return** $\sigma = (r, \mathbf{s})$

方式では, 連立方程式求解部を定数時間ガウスの消去法に変更し, 関数 Elim として導入した (Alg2. L14). 詳細は次節で後述する.

変更点 2 について, 現仕様版では, テーブル参照に基づく逆元導出の方法がとられているため, これに該当する関数がリーク箇所として検出されている. 逆元のテーブル参照は呼び出される回数がもともと少ないことを踏まえ本稿では, テーブル参照ではなく, 逆元を計算で求める方法に関数を書き換えることで定数時間処理とした.

また, 今回提案する方式は行列 L を正則と仮定している. 実際の署名生成では, L が正則でなかった場合に取り直すことで署名生成が可能となる.

4.2 連立方程式求解部分

QR-UOV の現仕様版では, 連立方程式求解部には LU 分解を用いたアルゴリズムが適用されており, LU 分解の際に起きる秘密情報に基づく行入れ替えの箇所が timecop_fail として検出されている. この対策として本稿では, LU 分解の代わりに定数時間ガウスの消去法 (Algorithm3) を適用する. 本手法は Rainbow 署名 [3] や UOV 署名 [2] の実装でも使用されているテクニックである. 一般的なガウスの消去法 (Algorithm4) と比較し, 定数時間ガウスの消去法の構成を説明する.

ガウスの消去法は主に前進消去と後退代入の二部構成

Algorithm 3 Constant-time linear equation solving using Gaussian elimination directly

Input: Linear equation $\mathbf{Ax} = \mathbf{b}$
Output: Solution $\mathbf{x}$ or $\perp$

1: $\mathbf{A}' := [\mathbf{A} \mid \mathbf{b}]$
2: **for** $i = 0$ upto $m - 1$ **do**
3: **for** $j = i + 1$ upto $m - 1$ **do**
4: **if** $a'_{i,i} == 0$ **then**
5: **for** $k = i$ upto m **do**
6: $a'_{i,k} := a'_{i,k} + a'_{j,k}$
7: **if** $a'_{i,i} == 0$ **then**
8: return $\perp$
9: $p_i^{-1} := (a'_{i,i})^{-1}$
10: **for** $k = i$ upto m **do**
11: $a'_{i,k} := p_i^{-1} \cdot a'_{i,k}$
12: **for** $j = i + 1$ upto $m - 1$ **do**
13: $\mathrm{mul} = a'_{j,i}$
14: **for** $k = i$ upto m **do**
15: $a'_{j,k} := a'_{j,k} - \mathrm{mul} \cdot a'_{i,k}$
16: **for** $i = m - 1$ downto 1 **do**
17: **for** $j = 0$ upto $i - 1$ **do**
18: $a'_{j,m} := a'_{j,m} - a'_{j,i} \cdot a'_{i,m}$
19: **return** last column of $\mathbf{A}'$

となるが, 従来のガウスの消去法では前進消去の前に, ピボット選択という操作が入ることが多い. ピボット選択とは, 行列の対角成分に 0 がでないよう, 事前に行の入れ替えを行う操作である. 具体的には, 列ごとに数値の評価を行い, 列の中で最も絶対値が大きい数値が対角成分に来るよう行の入れ替えを行う. 数値の絶対値の大きさ比較の操作が Algorithm4 の L5 ∼ 8 に該当し, 比較結果に基づく行入れ替えの操作は L11 ∼ 15 に該当する. 行のみを入れ替える場合を部分ピボット選択といい, 行と列を共に入れ替える場合を完全ピボット選択という.

定数時間ガウスの消去法では, ピボット選択 (行入れ替え) の代わりに, 一定の回数行加算を行うことで対角成分に 0 があらわれることを回避する方法をとる (Alg3. L3 ∼ 6). これにより, 秘密情報に基づく分岐を可能な限り抑えている.

しかし, Algorithm3 では, L4 と L7 の if 文でタイミングリークが起きるため, コード 1 で示す判定関数を使用する. 判定関数は入力が 0 の場合は 0 を返し, 0 以外の場合は 1 を返す関数である. この判定関数を用いて, 対角成分が 0 の場合は行加算を実施し, 0 以外の場合は 0 倍した行を加算する機構を作ることで, 対角成分に依らず行加算を行い, 条件分岐を回避している.

Algorithm 4 Linear equation solving using Gaussian elimination with pivoting

Input: Linear equation $\mathbf{Ax} = \mathbf{b}$
Output: Solution $\mathbf{x}$ or $\perp$

1: $\mathbf{A}' := [\mathbf{A} \mid \mathbf{b}]$
2: **for** $i = 0$ upto $m - 1$ **do**
3: $p_i = a'_{i,i}$
4: $pr = i$
5: **for** $j = i + 1$ upto $m - 1$ **do**
6: **if** $|p_i| < |a'_{j,i}|$ **then**
7: $p_i = a'_{j,i}$
8: $pr = j$
9: **if** $p_i == 0$ **then**
10: return $\perp$
11: **if** $i \neq pr$ **then**
12: **for** $k = 0$ upto m **do**
13: $p_i = a'_{i,k}$
14: $a'_{i,k} = a'_{pr,k}$
15: $a'_{pr,k} = p_i$
16: $p_i^{-1} := (a'_{i,i})^{-1}$
17: **for** $k = i$ upto m **do**
18: $a'_{i,k} := p_i^{-1} \cdot a'_{i,k}$
19: **for** $j = i + 1$ upto $m - 1$ **do**
20: $\mathrm{mul} = a'_{j,i}$
21: **for** $k = i$ upto m **do**
22: $a'_{j,k} := a'_{j,k} - \mathrm{mul} \cdot a'_{i,k}$
23: **for** $i = m - 1$ downto 1 **do**
24: **for** $j = 0$ upto $i - 1$ **do**
25: $a'_{j,m} := a'_{j,m} - a'_{j,i} \cdot a'_{i,m}$
26: **return** last column of $\mathbf{A}'$

コード 1: 判定関数

```
1 unsigned is_nonzero (unsigned a) {
2     unsigned r = (unsigned) 0 - a ;
3     r >>= INPUT_SIZE ;
4       // 入力値で立つビット分シフト
5     return r & 1 ;
6 }
```

4.3 提案方式の測定

先述した変更を加えた提案方式を dudect と TIMECOP で測定した.

dudect での測定結果を図 2 に示す. 測定の結果, 10^6 回までの測定では t 値が 5 を超えることなく 2 ∼ 2.5 弱の間で安定する結果となった. 現仕様版と比較して大きな変化はない.

TIMECOP での測定の結果, 現仕様で timecop_fail と出

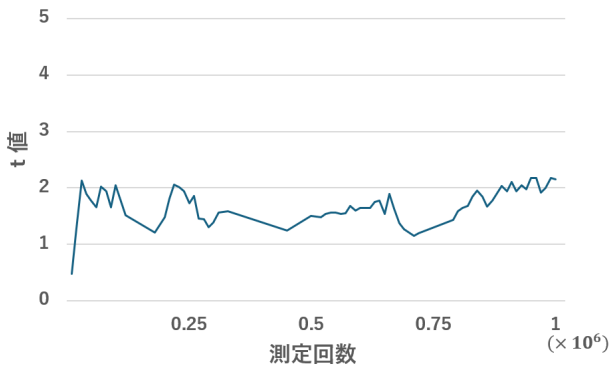

図 2: dudect による提案方式のタイミングリーク評価

力されていた箇所はすべて解消され, TIMECOP がタイミングリークを検出しなかったことを示す timecop_pass が出力された (表 1 参照).

表 1: TIMECOP による出力結果

	現仕様版	提案方式
連立方程式求解部	✓	
逆元計算	✓	

5 評価および考察

SUPERCOP での測定により, 処理速度 (サイクル数) の測定結果も得た. QR-UOV の現仕様版と提案方式の速度の変化は表 2 のようになった. 各セキュリティレベルからパラメータを一つずつ選び, 計三種類のパラメータで測定を行った結果, どれも概ね, 提案方式は現仕様版の 1.1 倍のサイクル数となり, これは定数時間処理によるサイクル数の増加分と考えられる.

表 2: 測定結果 (Mcycles)

カテゴリ	(q, v, m, ℓ)	現仕様版	提案方式
I	$(31, 165, 60, 3)$	35.243	39.104
III	$(31, 246, 87, 3)$	136.456	140.996
V	$(31, 324, 114, 3)$	406.660	411.438

本研究から得られた結果は次の通りである：

1. 現仕様において, 実行時間に大きなばらつきはないが、タイミングリーク箇所が検出されている.
2. 提案方式では, 性能を大きく低下させずにタイミングリーク箇所をすべてクリアしている.

各結果について考察も交えて説明をする. まず 1 について, QR-UOV の署名アルゴリズムは現仕様の時点で既に, t 値が 5 を超えることはなく, 2 ∼ 2.5 の幅で安定していることがわかった. 同様に提案方式も, t 値は 2 ∼ 2.5 弱の間で安定する結果となった.

2 について, 現仕様では連立方程式求解部, 逆元の計算, 棄却サンプリングの三か所が検出されていた. これらに対し, 脅威性の有無の判断と必要な箇所には定数時間処理を施したことで, 提案方式ではすべてクリアすることができた. また, 処理速度についても現仕様からの増え幅は 0.1 倍で抑えられていることも確認できた.

6 まとめ

本稿では, QR-UOV の定数時間実装の提案をした. 具体的には, まず現仕様版の測定をすることで具体的なタイミングリーク箇所を抽出し, 抽出した箇所すべてに対し脅威性の有無の判断および, 脅威性のある箇所については定数時間処理を施すことで, QR-UOV の定数時間実装を得た. 特に連立方程式求解部の解消について, アルゴリズムレベルで可能な限り秘密情報に基づく分岐を減らし, さらに実装レベルでの工夫を加えることで定数時間処理を実現した. 今後は, 部分測定などを行い定数時間処理による増加分の詳細を明らかにすることで, 性能向上 (高速化や定数時間処理の新規アイデアの検討) を行う.

参考文献

[1] Daniel J Bernstein and Tanja Lange. eBACS: ECRYPT benchmarking of cryptographic systems. 2009.

[2] 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. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):321–365, 2023.

[3] Tung Chou, Matthias J Kannwischer, and Bo-Yin Yang. Rainbow on cortex-m4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 650–675, 2021.

[4] Hiroki Furue, Yasuhiko Ikematsu, Fumitaka Hoshino, Tsuyoshi Takagi, Kan Yasuda, Toshiyuki Miyazawa, Tsunekazu Saito, and Akira Nagai. QR-UOV. *Specification document of NIST PQC Standardization of Additional Digital Signature Scheme*, 2023.

[5] Hiroki Furue, Yasuhiko Ikematsu, Yutaro Kiyomura, and Tsuyoshi Takagi. A new variant of unbalanced oil and vinegar using quotient ring: QR-UOV. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 187–217, Cham, 2021. Springer International Publishing.

[6] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

[7] Fumitaka Hoshino, Hiroki Furue, Yasuhiko Ikematsu, Tsunekazu Saito, Yutaro Kiyomura, and Tsuyoshi

Takagi. Efficient software implementation of signature scheme qr-uov. In *Symposium on Cryptography and Information Security (SCIS), 1A1-5*, 2023.

[8] 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.

[9] Paul Kocher. Differential power analysis. In *Proc. Advances in Cryptology (CRYPTO'99)*, 1999.

[10] Moritz Neikes. TIMECOP: Automated dynamic analysis for timing side-channels. 2019.

[11] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. *ACM Sigplan notices*, 42(6):89–100, 2007.

[12] NIST: Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process. `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf`, 2022.

[13] QR-UOV: Reference Implementation. `https://github.com/qruov/qruov`, 2023.

[14] Oscar Reparaz, Josep Balasch, and Ingrid Verbauwhede. Dude, is my code constant time? In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1697–1702. IEEE, 2017.

[15] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.