

QR-UOV の NIST PQC Round2 実装に対する定数時間実装の提案 Constant-Time Implementation for the NIST PQC Round2 QR-UOV

秋山 梨佳 *
Rika Akiyama

古江 弘樹 *
Hiroki Furue

金城 皓羽 *
Koha Kinjo

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 Round2 へ進出したことが発表されている. QR-UOV の Round2 実装は, 実行時間のばらつきは小さいものの, 完全な定数時間実装とはなっておらずタイミング情報が漏洩する可能性がある. 本稿では, タイミング攻撃の対象となる署名生成部について, その中でも特にタイミングリークのリスクのある連立方程式の求解処理に対し, 連立方程式のランク情報は公開情報とみなし, そのほかの秘密情報を秘匿する定数時間実装を提案する. その結果, 提案する実装では, より実行時間のばらつきを抑えられていること, また処理速度はオリジナルの実装の 1.2 倍の範囲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 [11] 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 [8].

デジタル署名方式 QR-UOV[2] 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 [1], 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 [9], 星野らによる効率的な実装手法 [4] 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 [8], 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, 現在公開されている QR-UOV の Round2 実装に対し, 定数

時間実装の提案をする. はじめに, QR-UOV の Round2 のリファレンスコードを解析し, 定数時間処理を施すべき部分を明らかにした. 次に, 洗い出したタイミングリークの懸念箇所に対し, マスク処理等の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. 最後に, 提案方式について性能測定を実施し, 実行時間のばらつきの観点から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. また, 本研究で提案する定数時間実装では, 定数時間化による処理性能の大幅な劣化を防ぐため, 連立方程式のランク情報は公開情報とみなし設計をしている. ランク情報が漏れることの脅威性についても検討をしておりその結果も報告する.

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, タイミングリーク測定ツールを用いた現在の実装の測定結果と, Round2 実装のコードを分析して得られたタイミングリークの懸念箇所の説明をする. また, 秘密情報の一つである連立方程式のランク情報を公開することでの安全性への影響について検討した結果を示す. 4 節では, 本稿で提案する定数時間実装の構成方針を示し, 具体的な定数時間処理について説明をする. 5 節では, 本稿で提案する定数時間実装に対し, タイミングリークと処理速度の測定をそれぞれ実施し, 得られた結果から考察をする.

* 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, hi-roki.furue, kouha.kinjo} @ntt.com)

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 $\mathcal{P} = (p_1, \dots, p_m) \in \mathbb{F}_q[x_1, \dots, x_n]^m$ が与えられたとき、 $\mathcal{P}(a_1, \dots, a_n) = 0 \in \mathbb{F}_q^m$ を満たす解 $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ 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 [3], 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 KeyGen(1^κ) $\rightarrow$ (sk, 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 Sign(sk, M) $\rightarrow \sigma$

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 Verify(M, σ, pk) $\rightarrow$ 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 KeyGen(1^κ) $\rightarrow$ (sk, pk)

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

2. 署名生成 Sign(sk, M) $\rightarrow \sigma$

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

3. 検証 Verify(M, σ, pk) $\rightarrow$ 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 [6] について説明する。 $\mathbf{x}_v := (x_1, \dots, x_v)$, $\mathbf{x}_o := (x_{v+1}, \dots, x_n)$, $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_o)$ 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 KeyGen(1^κ) $\rightarrow$ (sk, pk)

可逆な写像 $\mathcal{S} : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ をランダムに取る。また、 $\mathcal{F} = (f_1, \dots, f_m) : \mathbb{F}_q^n \rightarrow \mathbb{F}_q^m$ をとる。ただし各 f_k ($k = 1, \dots, m$) 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 Sign(sk, M) $\rightarrow \sigma$

署名鍵 sk を用いてメッセージ $M \in \mathbb{F}_q^m$ 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 $x_{v+1}, \dots, x_n$ 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 Verify(M, σ, pk) $\rightarrow$ 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 [2] 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:

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

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。

ここで、QR-UOV の公開鍵、秘密鍵に対する表現行列について定義しておく。秘密鍵、公開鍵の二次写像 $\mathcal{F} = (f_1, \dots, f_m)$, $\mathcal{P} = (p_1, \dots, p_m)$ の各二次多項式 f_i, p_i に対して、それぞれ表現行列 $F_i, P_i \in \mathbb{F}_q^{n \times n}$ を $f_i(x) = x^\top F_i x, p_i(x) = x^\top P_i x$ を満たす対称行列として定義す

る. また,

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

のように, 各表現行列の部分行列 $F_{i,1} \in \mathbb{F}_q^{v \times v}$, $F_{i,2} \in \mathbb{F}_q^{v \times m}$, $P_{i,1} \in \mathbb{F}_q^{v \times v}$, $P_{i,2} \in \mathbb{F}_q^{v \times m}$, $P_{i,3} \in \mathbb{F}_q^{m \times m}$ を定義する. また, 秘密鍵の線形写像 S に対して, $S(x) = Sx$ を満たすように行列 $S \in \mathbb{F}_q^{n \times n}$ を定義することができる. 行列 S は, 部分行列 $S' \in \mathbb{F}_q^{v \times m}$ を用いて,

$$S = \begin{pmatrix} I_v & S' \\ 0_{m \times v} & I_m \end{pmatrix},$$

のように構成するのが一般的である. このとき各 $i = 1, \dots, m$ に対して, 鍵生成の方法から,

$$P_i = S^\top F_i S,$$

が成り立つ. さらに, 各部分行列に注目すると,

$$\begin{aligned} P_{i,1} &= F_{i,1}, \\ P_{i,2} &= F_{i,1} S' + F_{i,2}, \\ P_{i,3} &= S'^\top F_{i,1} S' + F_{i,2}^\top S' + S'^\top F_{i,2}, \end{aligned}$$

が成り立つことがわかる.

2.2 タイミング攻撃と定数時間実装

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 年代後半から提案され始めた [7] 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) により様々な攻撃の種類がある. 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やループ回数の変動
(e.g. if $\langle \text{secret} == 0 \rangle$ then $\{\dots\}$ else $\{\dots\}$),
2. 秘密情報に依存するテーブル参照
(e.g. $s[i] = \text{substitution_table}[\text{secret}[i]]$),
3. 秘密情報により実行時間が変化する CPU 操作
(e.g. $x = \langle \text{secret} \rangle / y$).

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

3.1 リファレンス実装のタイミングリーク測定

本稿では, タイミングリークの測定方法として duedect[10] を用いる. duedect 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. 具体的には, 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. 本研究では, Goodwill et al. [5] により提案されている Welch の t 検定の閾値を参考とし, $|t| > 4.5$ の場合に統計的に有意な差があると判断した. また, 本稿での測定は, Intel Core i5-1245U (2.50 GHz), Ubuntu 22.04.5 を使用し, メモリ 16GB の環境で実施している.

dueduct によるリファレンス実装の署名生成部の測定結果を図 1, 2 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, 疑似乱数生成器として AES または SHAKE を使用した両方の場合において, 10^6 回までの実行では, t 値が 4.5 を超えることはなかった.

この結果より QR-UOV のリファレンス実装のタイミングリークについて, t 値の観点からは有意な差が観測されなかったことがわかるが, これは, リファレンス実装においてタイミングリークがないことを保証するわけではない. そのため, 実際に実装コードを分析しタイミングリークのリスク箇所に対し定数時間処理を施すことで, より強固なタイミング攻撃耐性を持たせる検討が必要である.

3.2 課題の整理

デジタル署名において, 主に署名生成部がタイミング攻撃の対象になる. そこで本節では, QR-UOV の Round2 リファレンス実装内の署名生成部に対し, コードを解析した結果を記載する.

署名生成部の中でも特に, 連立方程式の求解処理にタイミングリークの懸念がある. 連立方程式求解処理は主に 3 つの関数から構成されている. まず連立方程式の係

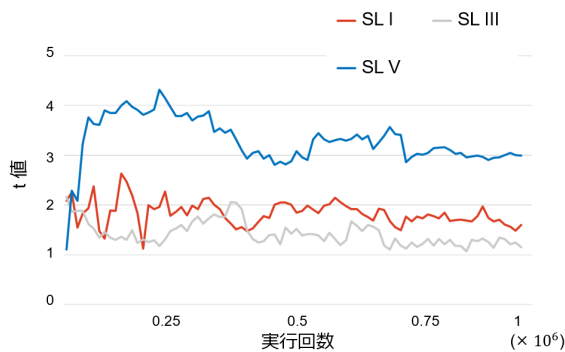


図 1: リファレンス実装のタイミングリーク評価 (AES を使用の場合)

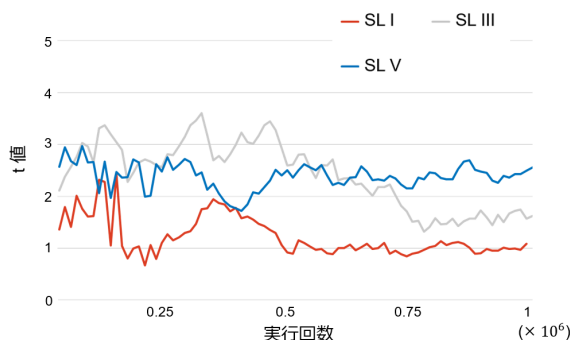


図 2: リファレンス実装のタイミングリーク評価 (SHAKE を使用の場合)

数行列を `LU_decompose` 関数で LU 分解し、解が存在するか否かを `consistent` 関数で判定し、解が存在する場合は `sample_a_solution` 関数で解を導出するという流れである。解析の結果、上記 3 つの関数は係数行列のランク情報に依る分岐やピボット選択による分岐を含むため処理時間が入力値依存となり、タイミングリークの可能性があることが分かった。各関数でのリーク要因の詳細については次の通りである。

1. `LU_decompose` 関数

`LU_decompose` 関数は、インプットされた行列に対し LU 分解を実施し、上三角行列と下三角行列をアウトプットする関数である。この関数での、タイミングリークのリスク懸念箇所は 2 つある。

1 箇所目は、ピボット探索部分での、対角成分の非ゼロ判定を実施する `while` ループ処理である。ここでは、行列要素の値に依存して探索回数変動するため、`pivot` 列や `pivot` 行に関する情報漏洩の可能性がある。実際、ピボット位置の候補が早期に見つかりループが早く終了し、対角要素に 0 があった場合は非ゼロ要素が見つかるまで探索を続ける機構となっている。

2 箇所目は、ランク情報に依存した早期終了の部分である。リファレンス実装では、連立方程式の係数行列について線形従属が早期に確定すると、`return` により関数が即時終了する機構となっているため、ランクの値が実行時間に反映される形となっている。特に、行列サイズを示すパラメータ m が大きい場合において差が顕著となり、ランク情報の観測可能性が高くなる。

2. `consistent` 関数

`consistent` 関数は、インプットされる行列とベクトルを用いて構成される連立方程式に対し、解の存在/非存在を判定する関数である。この関数では、解の存在/非存在の判定において、条件が不一致となった行が見つかり次第早期終了する機構となっているため、インプットされるベクトルのどの行で不整合が生じたかが実行時間に反映される可能性がある。これによりベクトルの構造やピボット列との関係などの情報がタイミング差として漏洩する可能性がある。

3. `sample_a_solution` 関数

`sample_a_solution` 関数は、インプットされる行列とベクトルから構成される連立方程式に対し、前進計算と後退計算を実施し解を求める関数である。この関数では、自由変数に対応する成分のみに対して乱数が割り当てられる機構となっているため、自由変数の位置やランクの大小に応じて乱数消費量が変動する。これにより、アウトプットされる解の構造を推定される可能性がある。

以上の分析から、現在の実装では、秘密情報であるピボット要素の位置情報、連立方程式のランク情報、連立方程式の右辺に当たるベクトルの情報、求解から得られる解ベクトルの構造の情報が実行時間の差異に反映される可能性があるため、これらの実装箇所に対し定数時間化を検討する必要があることがわかる。

3.3 ランク情報の安全性評価

前節で洗い出した定数時間化を施すべき箇所の中で、処理速度の劣化に最も影響を及ぼす可能性があるのがランク情報である。本節では、署名生成時に解く必要のある線型方程式の係数行列のランク情報が漏洩することが、安全性に与える影響について考察する。まず、ランク情報を利用した攻撃手法を提案し、その効率性をその他の攻撃手法と比較することで、ランク情報の漏洩が方式の安全性を損なわないことを主張する。

2.1 節で紹介した通り、QR-UOV 署名の署名生成においては、 $\mathcal{F}(x) = M$ の解 x を求める際に、vinegar 変数の

値をランダムに代入し, oil 変数についての線型方程式を求解する. このとき, vinegar 変数に代入する値 $\mathbf{x}_v$ に対して, 線型方程式の係数行列 $L(\mathbf{x}_v) \in \mathbb{F}_q^{m \times m}$ は

$$L(\mathbf{x}_v) = 2 \cdot \begin{pmatrix} \mathbf{x}_v^\top F_{1,2} \\ \vdots \\ \mathbf{x}_v^\top F_{m,2} \end{pmatrix}$$

として計算できる.

ここでは, 上で定義した行列 $L(\mathbf{x}_v)$ のランク情報を利用して効率的に鍵回復攻撃を行うことができるか考察する. ここで, 鍵回復攻撃とは, 公開鍵 P を入力として, 秘密鍵 S , もしくはその部分情報を復元する攻撃のことを指す. 今, 行列 $L(\mathbf{x}_v)$ を, 公開された情報である $P_1, \dots, P_m$ と署名 $\sigma = (\sigma_v, \sigma_o)$, 復元対象となる行列 S の部分行列 S' を用いて記述することを考える. 行列 $L(\mathbf{x}_v)$ の i 行目を $L_i(\mathbf{x}_v)$ とおくと,

$$\begin{aligned} L_i(\mathbf{x}_v) &= 2(\sigma_v + S' \sigma_o)^\top (-P_{i,1} S' + P_{i,2}) \\ &= 2\sigma_v^\top (-P_{i,1} S' + P_{i,2}) + 2\sigma_o^\top (-S'^\top P_{i,1} S' + S'^\top P_{i,2}) \\ &= 2\sigma_v^\top (-P_{i,1} S' + P_{i,2}) + 2\sigma_o^\top (-P_{i,2}^\top S' + P_{i,3}) \\ &= -2(\sigma_v^\top P_{i,1} + \sigma_o^\top P_{i,2}^\top) S' + 2(\sigma_v^\top P_{i,2} + \sigma_o^\top P_{i,3}), \end{aligned}$$

のように変形ができる. 上の式から, 行列 $L(\mathbf{x}_v)$ は変数 S' の要素に関する一次の多項式を成分にもつ行列とみなすことができる. さらに, $L(\mathbf{x}_v)$ の i 列目は, S' の i 列目の要素のみを含んだ多項式から構成されることがわかる.

攻撃者がタイミング攻撃によって, 署名 σ 生成時の行列 $L(\mathbf{x}_v)$ のランクが $r < m$ であるという情報を得たと想定する. 上述の変形から, S' の成分を変数として含んだ行列 $L(\mathbf{x}_v)$ に対して, ランクが r になるという条件を満たす S' を求めることで秘密鍵の復元が可能である. このとき攻撃者が解く問題は, 与えられた行列の集合からなる線型空間の要素のうちランクがある値以下になる要素を探す問題 (MinRank 問題) とみなすことができる. 以下がランク情報を利用した攻撃の手順である:

1. 署名オラクルにメッセージ M を複数回入力する.
2. 署名生成アルゴリズムの実行時間を解析し, $L(\mathbf{x}_v)$ のランクが $r < m$ となるメッセージと署名の組を見つける.
3. $L(\mathbf{x}_v)$ のランクが r となる条件を満たす S' を見つける.

次に, 実際のパラメータに対して, ランク r として取りうる値を見積もる. Round2 の security level I の推奨パラメータである $(q, v, m, l) = (127, 156, 54, 3)$ に対して本攻撃手法を適用することを考える. ここでは攻撃者

が署名生成オラクルを 2^{64} 回使用できるような状況を想定する. 行列 $L(\mathbf{x}_v)$ のランクが r になる確率をランダムに生成された行列の場合と等しいものとして計算すると, 該当のランクの行列 $L(\mathbf{x}_v)$ が得られる回数の期待値が 1 以上になるようなランクのうち最小のものは $r = 51$ の場合である.

最後に $L(\mathbf{x}_v)$ のランク情報を利用した攻撃と, 一般の鍵回復攻撃の効率性を比較する. 一般の鍵回復攻撃においては S' の一列ずつを対象とした復元が可能である. さらに QR-UOV に対しては, 剰余環構造を利用して拡大体 $\mathbb{F}_{q^l}$ 上の v/l 個の vinegar 変数を持つ UOV として鍵復元攻撃を実行可能である. このときパラメータ $(q, v, m, l) = (127, 156, 54, 3)$ に対して標準的な鍵回復攻撃の変数の個数は $156/3 = 52$ 個である. 一方で, ランク情報を利用して攻撃をする際には, ランクは少なくとも $r = 51$ であるので, $L(\mathbf{x}_v)$ の少なくとも 52 列を対象にして方程式を生成する必要がある. これは, 本攻撃手法においては S' の列のうち少なくとも 52 列を同時に復元する必要があることを意味する. また, $L(\mathbf{x}_v)$ は $\mathbb{F}_q$ 上の行列であり, そのランク情報を利用して構成される方程式も $\mathbb{F}_q$ 上のものになるため, 通常の鍵回復攻撃のように拡大体 $\mathbb{F}_{q^l}$ 上で攻撃を実行することはできない. よってパラメータ $(q, v, m, l) = (127, 156, 54, 3)$ に対するランク情報を利用した攻撃において, 変数の個数は $52v = 8112$ 個となる. これは, 通常の鍵復元攻撃時の変数の個数と比較して著しく大きい値である. また, 代数方程式の求解の計算量は主に変数の個数に依存することから, ランク情報を利用した攻撃は, 通常の拡大体上の鍵復元攻撃と比較して非効率であると考えられる. 以上の議論から, タイミング攻撃によって $L(\mathbf{x}_v)$ のランク情報が漏れた場合でも, 方式の安全性には影響を与えない.

4 提案

4.1 設計の方針

これまでの議論より本稿では, 連立方程式のランク情報は公開情報として扱い, それ以外のピボット要素の情報等は秘匿する方針の定数時間実装を提案する.

具体的には, LU 分解・解の存在判定・解の生成の 3 つの関数について, 以下の定数時間処理を施す:

1. 秘密情報に基づくループを固定回数にする,
2. 条件分岐をビットマスクによる条件付き代入に置き換える,
3. 値の計算を常に実行し, マスクで反映の有無を制御する,
4. 乱数は全成分に事前に割り当て, 必要な箇所は計算値で上書きする.

4.2 LU 分解関数の定数時間化

リファレンス実装での pivot 探索部分は、while ループで列ごとに最初の非ゼロ要素を探索する機構となっており、可変長の実行時間となっている。ここに対し、必ず全列×全行を走査し、各要素の非ゼロ判定結果に基づいてピボット候補をマスクで更新する機構に書き換えることで、固定回数での処理を実現する。具体的には、各列につき全行を走査する形とし、走査する各要素を 0 または 1 にビット変換し、その列で最初に非ゼロがあらわれた行をフラグを立てることで保持し、ピボット列の採用はマスクで更新するという機構に書き換える。これにより、pivot 探索の過程において、pivot 要素の列と行の情報が実行時間から推測されることを防止できる。

また、正規化および前進消去の計算部分については、全ての列に対し更新する計算を実行したうえで、pivot 列より右側にのみ計算結果を反映し、それ以外は元の値を保持する機構に書き換える。これにより pivot 列の位置に依存した実行時間の差異が生じることを防ぐことが可能になる。

4.3 解の存在判定関数の定数時間化

リファレンス実装では、不整合が起きた行が見つかり次第、関数が即時終了する機構となっている。ここに対し、判定のために必要な計算は全行で実施する形とし、各行での判定結果を論理演算で累積することで最終的な判定結果を得る機構に書き換える。この設計により、不整合がどの行で初めて起きたかに関する情報が外部に漏洩することを防ぐことができる。

4.4 解生成関数の定数時間化

リファレンス実装では、自由変数が入る箇所のみ乱数が割り当てられるため、乱数消費のパターンが自由変数の位置に直接対応する機構となっていた。ここに対し、解ベクトルの全要素に事前にランダム要素を割り当てておき、対応する箇所のみ後退計算で得られた値を上書きする機構に書き換える。これにより、pivot 構造に関する情報が秘匿できる。

また、関数内の後退代入の計算部分では、前述の前進消去の計算部分と同様に、pivot 列情報の秘匿のため、各列に対し、pivot 列か否かを示すマスクを生成し、対応する列のみ計算値を反映する機構に書き換える。

5 評価及び考察

先述した 3 つの関数への変更を施した提案方式を実装し、ducdet で測定した結果の一部を図 3 に示す。掲載している結果は、疑似乱数生成器として AES を使用した場合のセキュリティレベル V の推奨パラメータに対する、提案方式 (赤線) と現在の実装 (青線) の測定結果となっ

ている。提案方式について 10^6 回までの実行では、 t 値が 4.5 を超えることはなく 1~2 の幅で安定する結果となった。現在の実装の結果と比較し、1.5 ほどの t 値の低減が確認できる。また、セキュリティレベル I および III の推奨パラメータでも測定を実施し、提案方式においても t 値が 4.5 を超えることはなく、現在の実装に対する結果と概ね同値の幅で安定することも確認ができています。

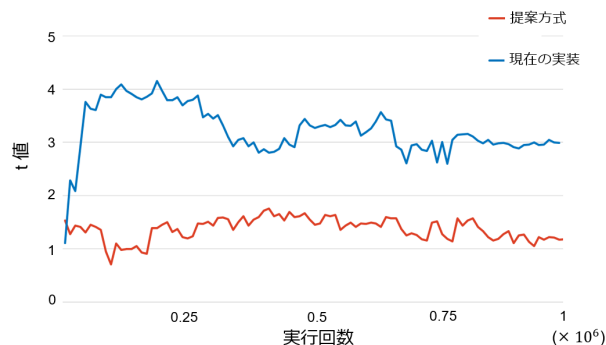


図 3: 現在の実装と提案方式のタイミングリーク評価

各セキュリティレベルでの推奨パラメータに対し、処理速度の測定を実施し、表 1, 2 の結果を得た。

表 1: 疑似乱数生成器に AES を用いた時の測定結果 (Mcycles)

カテゴリ	(q, v, m, ℓ)	現在の実装	提案方式
I	(127, 156, 54, 3)	2.55	3.20
III	(127, 228, 78, 3)	8.59	10.007
V	(127, 306, 105, 3)	21.58	25.04

表 2: 疑似乱数生成器に SHAKE を用いた時の測定結果 (Mcycles)

カテゴリ	(q, v, m, ℓ)	現在の実装	提案方式
I	(127, 156, 54, 3)	4.32	4.92
III	(127, 228, 78, 3)	14.88	16.39
V	(127, 306, 105, 3)	36.83	40.3

本研究から得られた結果は次の通りである：

1. QR-UOV の Round2 のリファレンス実装について、実行時間のばらつきに有意な差は見られなかったが、実装コードを分析した結果いくつかのタイミングリークのリスク箇所があった。
2. 秘密情報の一つであるランク情報について、公開情報としても安全性に影響を与えないことを確認し、ランク情報以外の秘密情報は保護する定数時間実装を提案した。提案方式では、性能を大きく低下させずに実行時間のばらつきがより抑えられていることを確認した。

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, duedect による測定の結果, QR-UOV の Round2 実装の署名生成部は, どの推奨パラメータにおいても, t 値が 4.5 を超えることはなく, 実行時間の有意な差は観測されなかった。しかし, 現在の実装は完全な定数時間実装とはなっておらず, 主にタイミング攻撃の対象となる署名生成部については, 特に連立方程式求解処理の部分にタイミングリークのリスク箇所がいくつかあることがわかった。

2 について, 連立方程式のランク情報を利用した攻撃の検討の結果, 通常の攻撃のように拡大体上で実行することができないため, 非効率となることがわかり, ランク情報を公開としても安全性に影響を与えないことを確認した。また, ランク情報以外の秘密情報は秘匿とする提案方式について, duedect での測定の結果, t 値は 4.5 を超えることはなく, リファレンス実装での t 値の遷移と概ね同様の結果を得た。その中でも特にセキュリティレベル V の推奨パラメータにおいて, 大幅な t 値の低減を確認できた。また, 処理速度について, 現在の実装からの処理速度の増え幅は 0.2 倍ほどで抑えられていることを確認した。

6 まとめ

本稿では, QR-UOV の Round2 実装に対し定数時間実装の提案をした。まず, 現在公開されている Round2 のリファレンス実装について, 定数時間実装はされていないが, 署名生成部の実行時間に有意な差は観測されなかった。しかしこの結果はタイミングリークのリスク箇所がないということを保証するものではない。そこで, 実際に実装コードを分析し, タイミングリークのリスク箇所を明らかにした。その中で, 定数時間化による大幅な性能劣化の可能性を考慮し, リスク箇所の一つである連立方程式のランク情報に依拠する実装部分に着目をした。ランク情報を公開情報とすることで方式の安全性に影響を与えないことを確認し, 本研究では, ランク情報は保護せず, それ以外の秘密の情報は秘匿する定数時間実装を提案した。実際に提案方式を実装・測定し, 一部のパラメータについて, より実行時間のばらつきが抑えられていることを確認した。また速度測定の結果から, 定数時間化による処理速度の増え幅は, 現在の実装のおよそ 0.2 倍となることを確認した。以上より, 安全性と実装コストのバランスを考慮した定数時間実装が実現できた。今後は, 安全性は保ちつつより性能が向上する実装方法の検討を進める。

参考文献

- [1] 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.

- [2] 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.
- [3] 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.
- [4] 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.
- [5] BJ Gillbert Goodwill J Jaffe and P Rohatgi. A testing methodology for side-channel resistance validation. https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf, 2011.
- [6] 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.
- [7] Paul Kocher. Differential power analysis. In *Proc. Advances in Cryptology (CRYPTO’99)*, 1999.
- [8] 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.
- [9] QR-UOV: Reference Implementation. <https://github.com/qr-uov/qr-uov>, 2023.
- [10] 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.
- [11] 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.