CONFERENCE PROCEEDINGS

Proceedings of the 9th International Cryptology and Information Security Conference 2024

24th - 26th September 2024
Cyberjaya, Selangor, MALAYSIA

Editors:
Muhammad Rezal Kamel Ariffin
Amir Hamzah Abd Ghafar
Muhammad Reza Z'aba
Wan Nur Aqlili Wan Mohd Ruzai
Zahari Mahad
Hazlin Abdul Rani

MSCR
Malaysian Society for Cryptology Research

CyberSecurity MALAYSIA

UPM
UNIVERSITI PUTRA MALAYSIA
BERILMU BERBAKTI

MIMOS

CRYPTOLOGY 2021

Proceedings of the 9th International Cryptology and Information Security Conference 2024

24th – 26th September 2024
Cyberjaya, Selangor,
Malaysia

Published by
Institute for Mathematical Research (INSPEM)
Universiti Putra Malaysia
43400 UPM Serdang
Selangor Darul Ehsan

First Print 2024

ISSN 2716-6783

EDITORIAL PREFACE

Since the time of Julius Caesar and possibly up until the Greek era, cryptography (a word that is derived from the Greek term “cryptos”) has been an integral tool for organizations (and indeed for individuals too) to ensure information that is intended only for authorized recipients remain confidential only to this set of people. Cryptography had far reaching implications for organizations in the event information leakage occurred. Often referred to as the “last bastion of defence" – after all other mechanisms had been overcome by an adversary, encrypted information would still remain useless to the attacker (i.e. that is, under the usual security assumptions). Nevertheless, this simple fact has remained oblivious to the practitioners of information security – omitting cryptographic mechanism for data being transferred and also during storage.

Fast forward to World War 2, – the war between cryptographic and cryptanalytic techniques. While the Germans were efficiently transferring information via the Enigma encryption machine, the Allies in Bletchley Park, England were busy intercepting these ciphered information being transmitted via telegraph by the Germans. Leading mathematicians, linguists, engineers etc. were all working to cryptanalyze these ciphers in the most information way. It is here that the first electrical machine (i.e. the “bomba”) was born – and revolutionized computing. Post World War 2 saw the emergence of the “computer”. Every organization that had to process data had to acquire a computer so as not to be left behind by their competitor. The banking sector advanced on a global scale due to the invention of the computer. Techniques to secure information among the headquarters of these banks had to be developed. Encryption procedures using the same key (i.e. symmetric encryption) played this role in the early days. Then came the unthinkable problem – computers were being deployed almost everywhere. How is it possible to deploy cryptographic keys in secure manner so that symmetric encryption could take place? Thus, leading to the so-called “key distribution” problem. It was not until 1975, when Diffie and Hellman provided us with a secure key exchange method – and in 1976 when Rivest, Shamir and Adleman with the “asymmetric encryption” scheme (i.e. to encrypt using key e and decrypt using key d, where $e \neq d$). Since then, cryptographic procedures evolved, not only playing the role of ensuring confidentiality of data, but also to ensure integrity and authenticity of data. It is also able to ensure that non-repudiating of data does not occur.

Mechanisms to transfer and store data has changed of the centuries and more so every 5 years (in this modern age). Cryptography that has long existed before mechanisms changed from manual – telegraphic – electrical – electronic (WAN/LAN/internet) – wired until wireless procedures, has to be properly deployed in order to maintain a high level of security confidence among the stakeholders of a certain organization. The concept

of securing information via encryption procedures has to be properly understood in order to avoid a null intersection to occur between cryptography and computer security practitioners. This scenario would not be to the best interest for stakeholders. As a "friendly" reminder, this scenario could already been seen in other discipline of knowledge where the "minuting" ("minute-ting") of knowledge has forced the original body of knowledge to look as though it is independent and disassociated. Ever since mass usage of computers became a reality, computer security issues have never been this complicated. However, as the human race advances so will ingenious ideas emerge to overcome challenges.

It is hoped that CRYPTOLOGY2024 will not only provide a platform for every participant to exchange ideas in their respective fields, but also to exchange new ideas on a broader scale for the advancement of the field of cryptology and computer security. The organizing committee hopes every participant will have an enjoyable and beneficial conference.

Thank you.

Editorial Board,
CRYPTOLOGY2024

ORGANISATION

General Chair	Amir Hamzah Abd Ghafar
International Program Committee	Amr M. Youssef
	Arif Mandangan
	Chin Ji Jian
	Chris Liaw Man Cheon
	Geong Sen Poh
	Hailiza Kamarulhaili
	Heng Swee Huay
	Kamel Ariffin Mohd Atan
	Mohd Anuar Mat Isa
	Muhammad Asyraf Asbullah
	Muhammad Reza Z'aba
	Nur Azman Abu
	Terry Lau
	Thomas Studer
	Yap Wun She
Executive Editors	Muhammad Rezal Kamel Ariffin
	Amir Hamzah Abd Ghafar
	Muhammad Reza Z'aba
	Wan Nur Aqlili Wan Mohd Ruzai
	Zahari Mahad
	Hazlin Abdul Rani
Technical Program Committee	Amir Hamzah Abd Ghafar
	Aniza Abd Ghani
	Hazlin Abdul Rani
	Muhammad Reza Z'aba
	Muhammad Rezal Kamel Ariffin
	Nik Azura Nik Abdullah

Committee Members	Faridatul Akhma Ishak
	Hazlin Abdul Rani
	Muhammad Asyraf Asbullah
	Muhammad Reza Z'aba
	Wan Nur Aqlili Wan Mohd Ruzai
	Zahari Mahad
Illustration & Art Work	Zahari Mahad

Table of Contents

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 1–11.

A Study on Multivariate Polynomial Solving in the Construction of Lattice-Based Cryptographic Signature: A Case of Signature Forgery

Nor Siti Khadijah Arunah[1,2] and Amir Hamzah Abd Ghafar[1,3]

[1] Institute for Mathematical Research, Universiti Putra Malaysia 43400, Selangor, Malaysia
[2] Department of Mathematics, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Cawangan Johor, 85000 Segamat, Malaysia norsi830@uitm.edu.my
[3] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia 43400, Selangor, Malaysia amir_hamzah@upm.edu.my

Abstract. Lattice-based and multivariate-based cryptosystems have an advantage in their relatively efficient encryption, decryption, and signing. The lattice-based hard problem depends on solving high-degree univariate polynomial in terms of $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ by finding the secret key $\mathbf{s}$ and error $\mathbf{e}$. For multivariate, the hard problem depends on the difficulty of solving m multivariate quadratic polynomial equations with n variable. This paper will show a few past successful attacks on a multivariate signature scheme. It succeeds in forging the signature using a rogue certificate attack by solving the polynomial of the multivariate cryptosystem's public key. We modified these attacks to enable them to attack lattice-based signature schemes. Moreover, we showed that solving the public key polynomial equation in a lattice-based signature scheme may require additional work before we retrieve its private keys.

Keywords: Lattice-based signature scheme · Learning with error · Post-quantum cryptography · Multivariate signature Scheme · Multivariate Quadratic Problem

1 Introduction

Regarding the development of the quantum computer, the National Institutes of Standards and Technology (NIST) has initiated a competition on post-quantum cryptography. In 2016, NIST announced receiving 23 signature schemes and 59 encryption or key encapsulation management (KEM) schemes into the competition [19]. Most submitted schemes focus on five mathematical approaches: multivariate-based, hash-based, code-based, supersingular elliptic curve isogeny-based, and lattice-based. In July 2022, NIST announced four cryptographic schemes [20] that have passed the first call of the six-year competition. In the list published, CRYSTALS-Kyber is a candidate under public-key encryption or key encapsulation mechanism (PKC/KEM), while CRYSTALS-Dilithium, FALCON, and SPHINCS+ are under digital signature schemes. Of the four algorithms, only SPHINCS+ is not lattice-based. This shows the importance of understanding lattice-based post-quantum cryptosystems.

Lattice-based cryptosystems use learning with error (LWE) as a hard problem. It requires several generations of secret randomness from discrete Gaussian distribution to build a system of linear equations with noise (more commonly known as "error"). The difficulties in solving (or learning) this "error-embedded" system of linear equations define lattice-based cryptography. The error or noise in the cryptographic system used in LWE will fail a simple row reduction method to solve it. Regev [21] first introduced the concept

of LWE in the lattice, used as a hard problem in cryptosystems by Güneysu et al. [13], followed by Bai and Galbraith [3], and adopted in the post-quantum cryptosystems in CRYSTALS-Dilithium [2] and FALCON [11] as signature schemes.

In particular, cryptosystems that use LWE as their hard problem will depend on $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ equation, where $\mathbf{A}$ is a uniformly random $k \times l$ matrix over $\mathbb{Z}_q$. Here, $\mathbf{A}$ is defined as a polynomial in the ring of $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $\mathbf{A} \in R_q^{k \times l}$. For vectors $\mathbf{s}$ and $\mathbf{e}$, it is defined as a small coefficient vector, with an element of $\mathcal{R}_q$ of size less than η.

For the multivariate-based cryptosystem, the security depends on the multivariate quadratic problem (MQP) first used by Garey and Johnson [12]. This method was applied to the Unbalanced Oil and Vinegar (UOV) scheme [16] and then to the Rainbow Signature Scheme [8]. The schemes are depending on the equation of $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ and this equation consists of an invertible quadratic map $\mathcal{F} : \mathbb{F}^m \to \mathbb{F}^m$ with another two invertible linear maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. MQP has three categories, which are underdetermined, determined, and overdetermined systems, depending on the size of m and n used in the system. Only an underdetermined system is suitable for application in building a signature scheme.

A digital signature adapts asymmetric cryptography to ensure the security and validation of any messages transmitted through an unsecured channel. It employed specific mathematical processes to authenticate digital documents whilst providing message integrity. Digital signatures also follow the goals of cryptography, confidentiality, data integrity, authentication, and non-repudiation. Forging a signature means manipulating the mathematical structure used in the cryptosystem to produce a valid signature. Here, multivariate-based and lattice-based signature schemes employ different mathematical procedures but involve the same structure: polynomials and matrices. Thus, in this work, we will compare both schemes to determine if the parameters provided by the rogue certificate provider will enable the forging of the signature in the case of multivariate-based and lattice-based signature schemes.

2 Related Works

Since the announcement of the post-quantum cryptosystem standardisation by NIST in 2016, many researchers have started working on attacking the post-quantum hard problems, including the hardness of multivariate and lattice problems. Here, we discuss previous results that aim to calculate solving polynomial parameters leading to forging the signature; the cases will compare lattice-based and multivariate-based signature schemes.

Miura et al. [18] emphasise in their research that to solve MQP is trying to get one solution of $(x_1, ..., x_n) \in k^n$ for equation $f_i(x_1, ..., x_n) = 0$ for all $i = 1, ..., m$ among all solution. The first to come out with an algorithm to solve this problem was Kipnis-Patarin-Goubin's algorithm [16], extended by Courtois et al.'s algorithm [6] and Thomae et al.'s algorithm [23]. Miura et al.'s proposed algorithm in [18] that solves the MQ-Problem for $f_i(x_1, ..., x_n) = \mathbf{x}F_i\mathbf{x} + (linear)$ where $F_i, ..., F_m$ are $n \times n$ matrices over k. In the research, they separated the quadratic polynomial from the linear polynomial. They solved the linear equations first where they let the linear equation $x_i L_{i,j}(x_{m+1}, \cdots, x_n) = 0$ and substitute the solution into the whole system and see if the solution match other conditions to consider the solution is valid. In this case, they stated that n, m must be greater than equal to 1 with $n \geq m(m+3)/2$.

Hashimoto in [14] solved the MQP for the underdetermined system of the case $n \geq m^2 - 2m^{3/2} + 2m$. His work stated that he needed to find an invertible matrix U that consists of an element $(u_{ij})_{0 \leq i,j \leq n}$, where if $u_{00} \neq 0$ and the coefficient of the highest degree, in this case, coefficient of x_0^2 in $f_l(U_x)$ is zero then the solution of the MQP problem is $(u_{00}^{-1}u_{10}, u_{00}^{-1}u_{20}, \cdots, u_{00}^{-1}u_{n0})$ for $f_1(x) = 0, f_2(x) = 0, \cdots, f_m(x) = 0$. In 2021, [4] improved the reconciliation attacks by [10] on the Rainbow Signature Scheme. A

multivariate signature scheme is an underdetermined system with the number of equations m less than the number of variable n or $m < n$. Beullens [4] attacked the Unbalanced Oil and Vinegar (UoV) signature scheme by separating the cases of the underdetermined system into two parts: if $n - m \leq m$ and $n - m > m$. He stated that the first case is considered a direct attack as calculating $\mathcal{P}(\mathbf{o}) = 0$ gives a unique solution. In the second case, calculating $\mathcal{P}(\mathbf{o}) = 0$ offers many solutions, with one of them corresponding to $\mathbf{o} \in O$.

In 2022, [1] constructed the strategies to forge a multivariate signature scheme by solving the multivariate polynomial in the system. In the paper, they show the Digital Signature Forgery Mechanism (DSFM) where they can solve the polynomial system or the signature is forgeable in 3 cases; DSFM1 show if $p^{(j)} = k_j p^{(1)}$ where $k_j \in \mathbb{Z}_q$; DSFM2 show if $p^{(j)} = p^{(i)} + p^{(k)}$ and DSFM3 show if $\mathcal{P}(\mathbf{x} + a) = \mathbf{w}$. For all these attacks, solving polynomials is needed, where finding x while $p(x) = 0$. In 2023, [15] showed in their works that the solution for $\mathcal{P}$ in a multivariate signature scheme is also the solution for the summation of all the polynomials in $\mathcal{P}$. In this paper, they also show that by solving the quadratic polynomial, they are solving the whole system if every polynomial in $\mathcal{P}$ a multiple of other polynomials such as $f_j(x) = kf_j(x)$ with k is a scalar. Here, if $\mathcal{P} = (f_1(x), ..., f_m(x))$, solving it by finding vector $\mathbf{x} = (x_1, ..., x_n)$ such that $p^{(1)}(\mathbf{x}) = ... = p^{(m)}(\mathbf{x}) = 0$.

For lattice-based signature schemes that adopt LWE as the hard problem, Regev [21] stated that solving LWE means finding the error or noise in the equation of $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ where $\mathbf{b}$ is a vector, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{s}$ is a small coefficient secret vector and $\mathbf{e}$ is a small coefficient vector denoted by the error or noise. CRYSTALS-DIlithium [2] and FALCON [11] adopt the variant of LWE hardness into the signature scheme where the use of high-degree univariate polynomials to strengthen the security, and we call the variant Module Learning with Error (MLWE) and Ring Learning with Error (RLWE).

2.1 Paper Organisation

Section 3 begins with an introduction to MQP and LWE, including the mathematical notations and the signing scheme process. Section 4 shows the method of solving the polynomial in MQP. In Section 5, we try to solve the polynomial in LWE by following the procedure of solving MQP and show if the solution solved the hard problem or led to forging the signature. Section 6 will be the conclusion of our work.

3 Preliminaries

This section will provide basics on the multivariate-based and lattice-based signing scheme hard problem and their key generation, signing and verification procedures based on UOV [16] and CRYSTALS-Dilithium [2] respectively.

3.1 Multivariate Quadratic Problem

The hard problem in MQP lies in identifying the vector $\mathbf{x} = (x_1, \cdots, x_n)$ in the system of m quadratic equations with n variables under the finite field $\mathbb{F}_q$ in the system $\mathcal{P}$. The system $\mathcal{P}$ consist of m quadratic equation which inscribed as $\mathcal{P} = \left(p^{(1)}, \cdots, p^{(m)}\right)$. One needs to recover vector $\mathbf{x}$ such that $\mathcal{P}(x) = 0$ to solve MQP and recover any secret key involved in the scheme.

From [7], the definition of quadratic polynomials in multivariate-based cryptography is as follows:

Definition 1. [Multivariate Quadratic Polynomial] Let $\mathbb{F} = \mathbb{F}_q$ be a finite field with q elements. The number of equations is denoted by m with n number of variables. The

system $\mathcal{P} = \left(p^{(1)}, \cdots, p^{(m)}\right)$ of multivariate quadratic polynomial is defines as

$$p^{(1)}(x_1, \cdots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(1)} \ldots x_i x_j + \sum_{i=1}^{n} p_i^{(1)} \ldots x_i + p_0^{(1)}$$

$$\vdots$$

$$p^{(m)}(x_1, \cdots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^{(m)} \ldots x_i x_j + \sum_{i=1}^{n} p_i^{(m)} \ldots x_i + p_0^{(m)}.$$

For a digital signature that uses MQP, we will use an underdetermined system, $m < n$, where the number of the equations is smaller than the number of variables.

3.1.1 Polynomial Representation

The hard problem of multivariate-based signature schemes depends on the equation of $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ and $\mathcal{P}$ is public while $\mathcal{S}$,$\mathcal{F}$ and $\mathcal{T}$ are kept secret. To solve this hard problem, one seeks to find and recover the three maps $\mathcal{S}$,$\mathcal{F}$ and $\mathcal{T}$ from the published public key $\mathcal{P}$.

In general, to generate public key $\mathcal{P}$ based on signature scheme UOV [16] and Rainbow [9], first, we let $\mathbb{F}_q$ be a finite field with q elements. The number of equations is o, and the number of variables is n, where $n = o + v$ and $v > o$.

Affine map ($\mathcal{S}$ and $\mathcal{T}$) and central map ($\mathcal{F}$) use in both scheme are in a quadratic polynomials $f^{(1)}, \cdots, f^{(o)}$ in the form below:

$$f^{(k)} = \sum_{a,b \in V} \alpha_{a,b}^{(k)} x_a x_b + \sum_{a \in V, b \in O} \beta_{a,b}^{(k)} x_a x_b + \sum_{a \in V \cup O} \gamma_a^{(k)} x_a + \delta^{(k)}$$

where $k = 1, \cdots, o$

From the equation, if $V = 1, \cdots, v$ and $O = v+1, \cdots, n$, then $x_1, \cdots, x_v$ are known as Vinegar variables while $x_{v+1}, \cdots, x_n$ are the Oil variables. After computing $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$, it will resulted polynomial for public key $\mathcal{P}$ as in Definition1.

3.1.2 The Basic Signature Scheme of Rainbow

This section shows the key generation process, signing and verifying a multivariate-based signature scheme, focusing on Rainbow [9].

Key Generation : Let $\mathcal{F}_q$ be a finite field with q elements and $v_1, \cdots, v_{u+1}$ be integers such that $0 < v_1 < \cdots < v_u < v_{u+1} = n$. Define the sets of integers $V_i = 1, \cdots, v_i$ for $i = 1, \cdots, u$ and set $o_i = v_{i+1} - v_i$ and $O_i = v_i + 1, \cdots, v_{i+1}$ for $i = 1, \cdots, u$. Here, $|V_i| = v_i$ and $|O_i| = o_i$. Generate the central map $\mathcal{F}$ that consists of $m = n - v_1$ polynomial. Next, generate two invertible affine maps $\mathcal{S}$ and $\mathcal{T}$ and composes $\mathcal{P}$ that consist of $\mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$. Hence, $\mathcal{P}$ is a public key and $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{T}$ is a private key.

Signature Generation : To generate the signature for message d, use the hash function to generate $\mathbf{W}$ where $\mathbf{w} = H(d)$ and compute $\mathbf{x} = \mathcal{S}^{-1}(\mathbf{w}) \in \mathbb{F}^m$. After that, we compute the pre-image $\mathbf{y} \in \mathbb{F}^n$ of $\mathbf{x}$ under the central map $\mathcal{F}$. Finally, compute the signature $\mathbf{z}$ which is $\mathbf{z} = \mathcal{T}^{-1}(\mathbf{y})$.

Signature Verification : To check if $\mathbf{z}$ is a valid signature, first compute $\mathbf{w} = H(d)$ and next compute $\mathbf{w'} = \mathcal{P}(\mathbf{z})$. If $\mathbf{w'} = \mathbf{w}$ accept the signature, otherwise reject.

3.2 Module Learning with Error

The LWE problem asks to recover the secret $\mathbf{s} \in \mathbb{Z}_q^n$ given the random linear equation system on $\mathbf{s}$. To illustrate it, suppose we have a system of the following equations

$$\begin{aligned} 7s_1 + 3s_2 + 4s_3 &= 4 \pmod 7 \\ 2s_1 + 10s_2 + 45s_3 &= 2 \pmod 7 \\ 14s_1 + 8s_2 + 12s_3 &= 4 \pmod 7 \end{aligned}$$

we can easily find the secret s_i for $i \in 1, 2, 3$ using Gaussian elimination polynomial time. Adding an error e_i for $i \in 1, 2, 3$ into the system will make the system hard to solve.

In the case of LWE, an error is formed in probability distributions $\chi \in \mathbb{Z}_q$, and we let $\mathbf{A}_{s,\chi}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ be the probability distribution obtained by choosing vector $\mathbf{a} \in \mathbb{Z}_q^n$. We then choose an error $\mathbf{e} \in \mathbb{Z}_q$ depending on χ and form a system of $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$, where the addition is performed in $\mathbb{Z}_q$. The equation is then equal to $(\mathbf{a}, \mathbf{b})$, which represents the public key of the LWE.

Meanwhile, MLWE was first studied by [5] and improved by [17]. In MLWE, $\mathcal{R}_q$ are polynomial in residual ring of degree n and modulus q and written as $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n+1)$. MLWE stated as matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$ and a ring vector $\mathbf{b}$ as

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$$

where $\mathbf{s} \in \mathcal{R}_q^l$ and $\mathbf{e} \in \mathcal{R}_q^k$ are small secret polynomial vector with $\mathbf{e}$ is the error added in the system.

The difference between LWE and its MLWE variant is that LWE uses large key sizes, although it is provably secure. Its variant MLWE offers a much more complicated algebraic structure, but it uses a small secret and error in which the coefficient used is bounded by η. In CRYSTALS-Dilithium, the secret key and error coefficient are bounded by centred η means the coefficient is in between $-\eta$ to η.

3.2.1 Polynomial Representation

From [21], the definition of the polynomial in lattice-based cryptography is as follows:

Definition 2. [Lattice-based Polynomials] Let $\mathcal{R}_q$ be a polynomial residual ring of degree n and modulus q with $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ and the matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$ and a ring vector $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ with the secret ring vector $\mathbf{s} \in \mathcal{R}_q^l$ and small secret error $\mathbf{e} \in \mathcal{R}_q^k$. Given a system $\mathcal{B} = (b^{(1)}(x), ..., b^{(k)}(x))$ of k univariate polynomial with degree n, find x such that

$$b^{(1)}(x) = ... = b^{(k)}(x) = 0.$$

A system $\mathcal{B} = (b^{(1)}(x), ..., b^{(k)}(x))$ of k univariate polynomial with degree n can also be defined as equation below:

$$\begin{aligned} b^{(1)}(x) &= \sum_{i=0}^{n} a_i^{(1)} x^i \\ &\vdots \\ b^{(k)}(x) &= \sum_{i=0}^{n} a_i^{(k)} x^i. \end{aligned}$$

Regev in [21] stated, solving LWE means we are being able to distinguish if vector $\mathbf{b}$ is just a random vector or if it is a combination of $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$. Our goal here is to find vector $\mathbf{e}$ or $\mathbf{s}$ from solving polynomial in $\mathbf{b}$. There are two versions of solving the LWE: search and decision. Search-LWE is to find $\mathbf{s}$ or $\mathbf{e}$ given polynomial matrix $\mathbf{A}$ and vector $\mathbf{b}$. In contrast, the Decision-LWE version distinguishes between an error inner product or a random sample from $\mathcal{R}_q^n$.

3.2.2 The Basic Signature Scheme of CRYSTALS-Dilithium

This section shows the key generation process, signing and verifying a lattice-based signature scheme, focusing on CRYSTALS-Dilithium [2].

Matrix $\mathbf{A}$ is generated in size $k \times l$, in which each entry in the matrix is a polynomial in the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ where $q = 2^{23} - 2^{13} + 1$ is a prime and n is fixed at 256. Secret vector $\mathbf{s}$ and $\mathbf{e}$ will then be generated randomly, with each of the coefficients being an element of $\mathcal{R}_q$ with a small size less than η.

Key Generation: Generate a polynomial matrix $\mathbf{A} \in \mathcal{R}_q^{k \times l}$, vector $\mathbf{s} \in \mathcal{S}_\eta^l$ and vector $\mathbf{e} \in \mathcal{S}_\eta^k$. Then compute $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ where $\mathbf{A}$ and $\mathbf{b}$ will be the public key while $\mathbf{s}$ and $\mathbf{e}$ be the private key.

Signature Generation: To generate the signature, we first have to generate a masking vector of polynomial $\mathbf{y} \in \mathcal{S}_{\gamma_1 - 1}^l$. Signer then compute $\mathbf{Ay}$ and find $\mathbf{w}_1$, where $\mathbf{w}_1$ is high-order bits of the coefficient in vector $\mathbf{Ay}$. Challenge c is created from the hash of $\mathbf{w}_1$ and message M. The signature is then computed as $\mathbf{z} = \mathbf{y} + c\mathbf{s}$. Next, rejection sampling ensures signature $\mathbf{z}$ will not leak any secret keys. Here, we set β to be the maximum possible coefficient of $c\mathbf{s}$, and check to ensure that $\mathbf{z}$ is less than $\gamma_1 - \beta$. If $\mathbf{z} > \gamma_1 - \beta$, then the second condition will run by finding the low order bits of coefficient $\mathbf{Az} - c\mathbf{b}$ and check if the low bits are less than $\gamma_2 - \beta$. If both conditions fail, then $\mathbf{z}$ is rejected, and the signing procedure will restart. This loop will continue until one of the two conditions is satisfied to ensure the scheme's security and correctness.

Signature Verification: Verifier first compute $\mathbf{Az} - c\mathbf{b}$ and set $\mathbf{w}_1'$ as the higher order bits. Accept the signature if the coefficient of all polynomials in vector $\mathbf{z}$ are less than $\gamma_1 - \beta$ and c is the hash of the message M concatenated with $\mathbf{w}_1'$.

This verification actually works because $\mathbf{Az} - c\mathbf{b}$ is actually equal to $\mathbf{Ay} - c\mathbf{e}$. These algorithms will then show that Highbits($\mathbf{Ay}$) are equal to Highbits($\mathbf{Ay} - c\mathbf{e}$). For the detailed procedure, please refer to [2].

3.3 Comparison of Multivariate-based and Lattice-based Signature Scheme

Here, we made a detailed comparison between both signature schemes, in the case of Rainbow, which is multivariate-based, and CRYSTALS-Dilithium, which is a lattice-based signature scheme.

Table 1: Comparison between Rainbow with CRYSTALS-Dilithium Signature Scheme

Comparison	Multivariate-based (Rainbow)	Lattice-based (CRYSTALS-Dilithium)
Hard Problem	Multivariate Quadratic Problem	Learning with Error
Trap-door functions property	Hard to invert	Hard to solve
Mathematical structure	Polynomial matrix	Polynomial matrix
Polynomial indeterminate	Multivariate	Univariate
Degree of polynomial	2	256

Table 1 shows that both signature schemes use the same mathematical structure, a polynomial matrix, and a lot of attack and signature forgery is done for the Rainbow signature scheme, which is already discussed in Section 2. Comparing between multivariate-based and lattice-based signature schemes on their polynomials and matrices, we observe

that a lattice-based signature scheme has a vector $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$,

$$\mathbf{b}_{k,1} = \begin{pmatrix} b_{1,1}(x) \\ b_{2,1}(x) \\ \vdots \\ b_{k,1}(x) \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n-1} b_i^{(1)} x^i \\ \sum_{i=0}^{n-1} b_i^{(2)} x^i \\ \vdots \\ \sum_{i=0}^{n-1} b_i^{(k)} x^i \end{pmatrix} \tag{1}$$

and for the multivariate signature scheme, we let $\mathcal{P}$ as a multivariate quadratic system consisting of m polynomial equation with n variable over $\mathbb{F}_q$. Thus, $\mathcal{P}$ can be written as,

$$f_1(\mathbf{x}) = f_1(x_1, \cdots, x_n)$$

$$f_2(\mathbf{x}) = f_2(x_1, \cdots, x_n)$$

$$\vdots$$

$$f_m(\mathbf{x}) = f_m(x_1, \cdots, x_n)$$

that can be written in matrix form as,

$$\mathcal{P} = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, \cdots, x_n) \\ f_2(x_1, \cdots, x_n) \\ \vdots \\ f_m(x_1, \cdots, x_n) \end{pmatrix} \tag{2}$$

and we should have the set $\mathbf{x} = (x_1, \cdots, x_m)$ as the solution of the multivariate system.

From equations (1) and (2), we can see some similarities in the polynomial equations used in the key generation processes of lattice-based and multivariate-based signing schemes, which we hope to manipulate in the proceeding sections.

4 Solving for Polynomial in MQP

From equation (2) we can see that $\mathcal{P} = (f_1(\mathbf{x}), \cdots, f_m(\mathbf{x}))$ is s system of m multivariate quadratic. [1] stated in their research paper that, to forge the multivariate signature, we need to either generate the valid signature **s'** where $\mathcal{P}(\mathbf{s'}) = \mathbf{z'} = \mathbf{z}$ or we can find the secret key $\mathcal{S}, \mathcal{F}$ and $\mathcal{T}$.

Jamal et al. [15] make two feasible attacks on a Multivariate-based signature scheme. To summarise the attack, [15] proves that the signature can be forged if they can solve the polynomial equation in the public key. In all three attacks, they show that $f_1(\mathbf{x}) = \cdots = f_m(\mathbf{x}) = 0$ and find $\mathbf{x} = (x_1, \cdots, x_n)$ as the solution of all equations. In a multivariate-based signature scheme, $\mathcal{P}$ is considered forgeable if the polynomial equation in $\mathcal{P}$ is solved.

The two feasible attacks done in the research depend on the variation of polynomials in $\mathcal{P}$ where the first attack is polynomials in $\mathcal{P}$ are written in $f_j(\mathbf{x}) = k_j f_1(\mathbf{x})$ where $j = 2, \cdots, m$. From here, if $f_1(\mathbf{x}) = z_1$ then for k_j, it will became as follows,

$$f_2(\mathbf{x}) = z_2 = k_2 z_1$$

$$\vdots$$

$$f_m(\mathbf{x}) = z_m = k_m z_1$$

from here, we can conclude that the polynomial solution in [15] is also the solution of other multivariate quadratic polynomials in the system $\mathcal{P}$ if the polynomials are multiple of each other.

It is the same as the second attack, [15] stated that $\mathcal{P}$ is also the solution for the summation of every polynomial in a system $\mathcal{P}$. In detail, they stated that if we have

$$f_h(\mathbf{x}) = f_1(\mathbf{x}) + \cdots + f_m(\mathbf{x})$$

and we found the solution for all $x_1, \cdots, x_n$ by solving for $f_h = 0$ and the whole system is solved.

5 Solving for Polynomial in MLWE

We start with a matrix $\mathbf{A}$ in the system of CRYSTALS Dilithium. There are three sizes of matrix $\mathbf{A}$ according to [2], which are $4 \times 4, 6 \times 5$ and 8×7 depending on the NIST security level of 2,3 and 5, respectively.

$$\mathbf{A}_{k,l} = \begin{pmatrix} A_{1,1}(x) & A_{1,2}(x) & \cdots & A_{1,l}(x) \\ A_{2,1}(x) & A_{2,2}(x) & \cdots & A_{2,l}(x) \\ \vdots & \vdots & \ddots & \vdots \\ A_{k,1}(x) & A_{k,2}(x) & \cdots & A_{k,l}(x) \end{pmatrix}$$

Matrix $\mathbf{A}$ consist of $k \times l$ polynomial equation with degree at most n in every equation following ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. We can see that each polynomial equation in $\mathbf{A}$ are as follows:

$$A_{1,1}(x) = \sum_{i=0}^{n-1} a_i^{(1,1)} x^i$$

$$A_{1,2}(x) = \sum_{i=0}^{n-1} a_i^{(1,2)} x^i$$

$$\vdots$$

$$A_{k,l}(x) = \sum_{i=0}^{n-1} a_i^{(k,l)} x^i$$

Since LWE equations are built based on $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, observe the secret vector $\mathbf{s}$ and $\mathbf{e}$. Vector $\mathbf{s}$ and $\mathbf{e}$ consist of $l \times 1$ and $k \times 1$ polynomial equations with degree at most n, respectively.

$$\mathbf{s}_{l,1} = \begin{pmatrix} s_{1,1}(x) \\ s_{2,1}(x) \\ \vdots \\ s_{l,1}(x) \end{pmatrix} = \begin{pmatrix} \sum_{j=0}^{n-1} s_j^{(1)} x^j \\ \sum_{j=0}^{n-1} s_j^{(2)} x^j \\ \vdots \\ \sum_{j=0}^{n-1} s_j^{(l)} x^j \end{pmatrix}$$

$$\mathbf{e}_{k,1} = \begin{pmatrix} e_{1,1}(x) \\ e_{2,1}(x) \\ \vdots \\ e_{k,1}(x) \end{pmatrix} = \begin{pmatrix} \sum_{g=0}^{n-1} e_g^{(1)} x^g \\ \sum_{g=0}^{n-1} e_g^{(2)} x^g \\ \vdots \\ \sum_{g=0}^{n-1} e_g^{(k)} x^i \end{pmatrix}$$

After we compute $\mathbf{b}$ which is equal to $\mathbf{A} \cdot \mathbf{s} + \mathbf{e}$, $\mathbf{b}$ need to be reduced following the ring $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$. We will get every polynomial in $\mathbf{b}$ as follows:

$$\mathbf{b}_{k,1} = \begin{pmatrix} b_{1,1}(x) \\ b_{2,1}(x) \\ \vdots \\ b_{k,1}(x) \end{pmatrix} = \begin{pmatrix} \sum_{i=0}^{n-1} b_i^{(1)} x^i \\ \sum_{i=0}^{n-1} b_i^{(2)} x^i \\ \vdots \\ \sum_{i=0}^{n-1} b_i^{(k)} x^i \end{pmatrix}$$

or

$$b_{1,1}(x) = [A_{1,1}(x) \cdot s_{1,1}(x) + A_{1,2}(x) \cdot s_{2,1}(x) + \cdots + A_{1,l}(x) \cdot s_{l,1}(x)] + e_{1,1}(x)$$

$$\vdots$$

$$b_{k,1}(x) = [A_{k,1}(x) \cdot s_{1,1}(x) + A_{k,2}(x) \cdot s_{2,1}(x) + \cdots + A_{k,l}(x) \cdot s_{l,1}(x)] + e_{k,1}(x)$$

$$b_{1,1}(x) = \left[\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(1,1)} s_j^{(1)} x^{i+j} + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(1,2)} s_j^{(2)} x^{i+j} + \cdots + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(1,l)} s_j^{(l)} x^{i+j} \right] + \sum_{g=0}^{n-1} e_g^{(1)} x^g$$

$$\vdots$$

$$b_{k,1}(x) = \left[\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(k,1)} s_j^{(1)} x^{i+j} + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(k,2)} s_j^{(2)} x^{i+j} + \cdots + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i^{(k,l)} s_j^{(l)} x^{i+j} \right] + \sum_{g=0}^{n-1} e_g^{(k)} x^g$$

The multiplication of two polynomials from the matrix $\mathbf{A}$ with vector $\mathbf{s}$ which is $\mathbf{As}$ will raise the degree of the solution exceeding the maximum degree. For CRYSTALS-Dilithium, the highest degree in the solution should not be greater than n; thus, each degree of the polynomial in vector $\mathbf{b}$ will be reduced after we use the Number Theoretic Transform or NTT [22] to multiplying matrix $\mathbf{A}$ with vector $\mathbf{s}$. Besides, every coefficient in the equation is also reduced following the ring $\mathcal{R}_q$ set earlier following the algorithm in [3].

We then solved the polynomial in $\mathbf{b}$. We first let $b_{(1,1)}(x) = b_{(2,1)}(x) = \cdots = b_{(k,1)}(x) = 0$ and then solve for x. Each equation in $\mathbf{b}$ will give different x values, and each equation will have at most $n-1$ values. This happens because lattice-based cryptography uses univariate, with $n-1$ degree of polynomial at most. But, if we let the structure of LWE be multiple of the first equation, which means $b_{m,1}(x) = f_m b_{1,1}(x)$ where $m = 2, \cdots, k$, then the solution of the first equation in $\mathbf{b}$ are the solution of all equation in $\mathbf{b}$. If we can write all of the polynomial equations in $\mathbf{b}$ as

$$b_{2,1}(x) = f_2 b_{1,1}(x)$$

$$\vdots$$

$$b_{k,1}(x) = f_k b_{1,1}(x)$$

where $f_m \in \mathbb{Z}_q$ we will get the solution for x such that $b_{1,1}(x) = 0$. To illustrate, when we let $\mathbf{b} = 0$, and if the polynomial can be factored, then it will give a solution for x,

$$b_{1,1}(x) = \sum_{i=0}^{n-1} b_i^{(1)} x^i = (x - b_1)(x - b_2) \cdots (x - b_{n-1})$$

x will equal to $b_1, b_2, ..., b_{n-1}$ or simply said that x will have at most $n-1$ solution to be analyse. Then

$$b_{2,1}(x) = f_2 b_{1,1}(x) = f_2(0) = 0$$

$$\vdots$$

$$b_{k,1}(x) = f_k b_{1,1}(x) = f_k(0) = 0.$$

We conclude here if we can solve for x for the first polynomial in $\mathbf{b}$ where we let $\mathbf{b}(x) = 0$ then and we have also solve for x in the whole system of LWE. However, the solution for x itself is insufficient to solve the whole equation and extract the secret key.

6 Conclusion

In conclusion, following the CRYSTALS-Dilithium signature scheme procedure, we compared the approach of solving multivariate signature schemes that use MQP as their hard problem that mainly focused on the case of Rainbow and UOV with solving the polynomial of the lattice-based signature scheme. Here, we identified that solving a lattice-based signature scheme's high-degree polynomial system will not be enough to retrieve the private keys. In this paper, we use two approaches: finding the unknown and solving the polynomial by manipulating the equation. Both approaches show promise and require further development to effectively solve the system or address signing forgery. However, factoring the polynomial in the system is possible despite having a lot of potential solutions, knowing that CRYSTALS-Dilithium polynomials are in a high degree. In the multivariate signature schemes, some parameters were provided that allowed for the forgery of the digital signature. For the cases discussed above, we recommend focusing on the NTT process of addition and multiplication, which makes it easier to manipulate the system that might lead to signature forgery.

References

[1] Nurul Amiera Sakinah Abdul Jamal, Muhammad Rezal Kamel Ariffin, Siti Hasana Sapar, and Kamilah Abdullah. New Identified Strategies to Forge Multivariate Signature Schemes. *Symmetry*, 14(11):2368, 2022.

[2] Shi Bai, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (version 3.1). *NIST Post-Quantum Cryptography Standardization Round*, 3, 2021.

[3] Shi Bai and Steven D Galbraith. Lattice Decoding Attacks on Binary LWE. In *Information Security and Privacy: 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings 19*, pages 322–337. Springer, 2014.

[4] Ward Beullens. Improved Cryptanalysis of UOV and Rainbow. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 348–373. Springer, 2021.

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[6] Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving Underdefined Systems of Multivariate Quadratic Equations. In *International Workshop on Public Key Cryptography*, pages 211–227. Springer, 2002.

[7] Jintai Ding, Albrecht Petzoldt, and Dieter S. Schmidt. *Multivariate Cryptography*, pages 7–23. Springer US, New York, NY, 2020.

[8] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 164–175. Springer, Heidelberg, June 2005.

[9] Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, pages 164–175, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[10] Jintai Ding, Bo-Yin Yang, Chia-Hsin Owen Chen, Ming-Shing Chen, and Chen-Mou Cheng. New Differential-algebraic Attacks and Reparametrization of Rainbow. In *Applied Cryptography and Network Security: 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings 6*, pages 242–257. Springer, 2008.

[11] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process*, 36(5):1–75, 2018.

[12] Michael R Garey, David S Johnson, et al. A Guide to the Theory of NP-Completeness. *Computers and intractability*, pages 37–79, 1990.

[13] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical Lattice-based Cryptography: A Signature Scheme for Embedded Systems. In *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*, pages 530–547. Springer, 2012.

[14] Yasufumi Hashimoto. Algorithms to Solve Massively Under-defined Systems of Multivariate Quadratic Equations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(6):1257–1262, 2011.

[15] NASA Jamal and Muhammad Rezal Kamel. Novel Forgery Mechanisms in Multivariate Signature Schemes. *Comput. Sci*, 18(3):451–461, 2023.

[16] 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.

[17] Adeline Langlois and Damien Stehlé. Worst-case to Average-case Reductions for Module Lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[18] Hiroyuki Miura, Yasufumi Hashimoto, and Tsuyoshi Takagi. Extended Algorithm for Solving Underdefined Multivariate Quadratic Equations. In *Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5*, pages 118–135. Springer, 2013.

[19] Dustin Moody. The Future Is Now: Spreading the Word About Post-Quantum Cryptography. *URl: https://www.nist.gov/blogs/taking-measure/future-now-spreading-word-about-post-quantum-cryptography*, 2020.

[20] PQC Standardization Process. Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates. *URl: https://csrc. nist. gov/News/2022/pqc-candidates-to-be-standardized-and-round-4# newcall*, 2022.

[21] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[22] Michael Scott. A Note on the Implementation of the Number Theoretic Transform. In *Cryptography and Coding: 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings 16*, pages 247–258. Springer, 2017.

[23] Enrico Thomae and Christopher Wolf. Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In *International workshop on public key cryptography*, pages 156–171. Springer, 2012.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 12–24.

Memory-Efficient Implementations of CRYSTALS-Kyber

Nascimo Madieta[1,2], Guillaume Aymard[2], Nadia El Mrabet[1] and Daniele Fronte[2]

[1] SAS - CGCP - Mines Saint Etienne - France
Firstname.lastname@emse.fr
[2] SEALSQ
{nmadieta,gaymard,dfronte}@@SEALSQ.com

Abstract.
This paper presents a software memory-efficient implementation of CRYSTALS-Kyber that can be deployed on standard micro-controllers and embedded devices. The implementation is cross-platform and can be integrated into products already existing in the field. The goal is to reduce the memory footprint of the CRYSTALS-Kyber execution while maintaining performance similar to the reference implementation[1]. The work results in two different implementations: one with a low memory footprint and stable speed performance and one with a memory-time trade-off option. In comparison with the 20KB memory requirement of the reference implementation, one of our implementations utilises less than 3KB of memory for each of these algorithms. Both implementations merge all Kyber variants (Kyber-512, Kyber-768, Kyber-1024) into a single code and adhere to NIST implementation recommendations, as outlined in the FIPS-203 directive. The perspective offers several possibilities to improve the implementation's speed.

Keywords: Post-quantum cryptography · CRYSTALS-Kyber · memory-efficient implementation · software implementation · cross-platform embedded implementation · ARM · Risc-V

1 Introduction

With the advancements in quantum computer research [7], classical asymmetric cryptography (RSA, ECC, etc.) appears to be under threat. It is believed that a quantum computer can break classical asymmetric cryptography by utilizing Shor's algorithm [12]. In 2016, the National Institute of Standards and Technology (NIST) announced a standardization process for post-quantum cryptography. The goal of this process was to select a cryptographic algorithm that will be resistant to quantum computers in the future. After six years, three elimination rounds, and 69 different proposals, NIST selected four algorithms CRYSTALS-Kyber, CRYSTALS-Dilithium, FALCON and Sphincs+ with the intention of standardizing them. These algorithms were selected after significant investment and effort from the scientific community. The only key encapsulation mechanism (KEM) in this list is CRYSTALS-Kyber. CRYSTALS-Kyber submission [3] was selected for its fast key generation, encapsulation, and decapsulation [13]. It also uses a small public key, a small secret key, and a small cipher text. So, Kyber has excellent performance in many settings. Kyber's security is based on the Module Learning With Errors (MLWE) problem [8], which is believed to be secure against quantum adversaries.

[1] https://github.com/pq-crystals/Kyber/tree/main

Contribution

This work presents two software optimizations to reduce the random-access memory (RAM) footprint in Kyber implementation with FIPS (Federal Information Processing Standard) parameters. We consider two approaches:

1. one with stable speed performance;
2. another with a memory-time trade-off for very constrained devices, such as smart card.

We restrict ourselves to algorithmic and C-language approach for implementing Crystals-Kyber in existing field-deployed products. The key benefit lies in its compatibility with a wide range of platforms, making it suitable for adoption across diverse systems. Our implementation avoids reliance on architecture-specific methods, ensuring robustness and portability. Our investigation involved various embedded platforms, including ARM and RISC-V architectures. We successfully unified multiple security variants into a single cohesive codebase while maintaining cross-platform compatibility. This consolidation streamlines maintenance, enhances security, and simplifies deployment across heterogeneous environments. We also discuss the difference between FIPS Module Lattice-based (ML)-KEM and CRYSTALS-Kyber's version and update our code with the NIST standardization guideline. At the end, we suggest several perspectives for future research.

Organization of the paper

Section 2 describes the CRYSTALS-Kyber key encapsulation mechanism and the difference with the NIST standardization. Section 3 presents the algorithm optimizations and the implementation optimizations. Section 4 presents the performance results of our software implementations. In Section 5, we conclude with various ways to improve Kyber's implementation and reference studies in this area.

2 Kyber

Several versions of the Kyber key encapsulation algorithm exist. This section provides an overview of the Kyber submission and algorithms [3].

2.1 Notation

Let q be a prime integer and n be a power of two. In the following, we use regular lower-case letters p for polynomials in the ring $\mathcal{R}_q = \mathbb{Z}_q/(X^n+1)$, where $\mathbb{Z}_q$ is a $\mathbb{Z}/q\mathbb{Z}$ field. Polynomials in Number-Theoretic Transform (NTT) representation are denoted $\hat{p}$. We also use bold lower-case letters $\mathbf{v}$ for vectors and bold upper-case letters $\mathbf{M}$ for matrices. So, vectors and matrices are both over $\mathcal{R}_q$ and $\mathcal{T}_q$ (the NTT domain of $\mathcal{R}_q$ elements). We denote by $\circ$ the coefficient-wise multiplication of two polynomials (respectively matrix coefficient-wise multiplication vector) in the NTT domain (the algorithms are built to avoid multiplication on $\mathcal{R}_q$, all multiplications are intended to happen in the NTT domain). Let $\hat{\mathbf{a}} \in \mathcal{T}_\mathbf{q}^\mathbf{k}$, $\hat{\mathbf{b}} \in \mathcal{T}_\mathbf{q}^\mathbf{k}$ so $\hat{\mathbf{a}} \circ \hat{\mathbf{b}} = \hat{\mathbf{c}}_\mathbf{1} \in \mathcal{T}_\mathbf{q}^\mathbf{k}$ and let $\hat{\mathbf{A}} \in \mathcal{T}_\mathbf{q}^{\mathbf{k}\times\mathbf{k}}$, $\hat{\mathbf{b}} \in \mathcal{T}_\mathbf{q}^\mathbf{k}$ so $\hat{\mathbf{A}} \circ \hat{\mathbf{b}} = \hat{\mathbf{c}}_\mathbf{2} \in \mathcal{T}_\mathbf{q}^\mathbf{k}$. Lower-case Greek ρ, σ and μ refer to random bit-strings that are used as seed.

2.2 Specification

Kyber is a post-quantum KEM, used for securely exchanging a *shared secret* between two parties over a public channel. Kyber originated from the LPR encryption scheme (Lyubashevsky, Peikert, and Regev), which was introduced in 2010 [3, 10]. The module

Table 1: Kyber parameter sets

	n	k	q	η_1	η_2	(d_u,d_v)	δ
KYBER512	256	2	3329	3	2	(10,4)	2^{-139}
KYBER768	256	3	3329	2	2	(10,4)	2^{-164}
KYBER1024	256	4	3329	2	2	(11,5)	2^{-174}

learning with error problem hardness [3, 8] serves as the foundation for this. Kyber is a specific instance of the learning with error problem. Solving a noisy system of equations is a hard problem. Let a matrix $\mathbf{A} \in \mathcal{R}_{\mathbf{q}}^{\mathbf{k}\times\mathbf{k}}$ and two vectors $\mathbf{s}$, $\mathbf{e} \in \mathcal{R}_{\mathbf{q}}^{\mathbf{k}}$ (with some properties), the noisy system of equations $\mathbf{A} \times \mathbf{s} + \mathbf{e} = \mathbf{z}$ seems hard to solve, given ($\mathbf{A}$, $\mathbf{z}$) and finding ($\mathbf{s}$, $\mathbf{e}$). More in details, vectors $\mathbf{s}$ and $\mathbf{e}$ are sample from a centered binomial distribution over the support $\{-\eta_i, \ldots, \eta_i\}$, $i \in \{1, 2\}$. Kyber uses $q = 3329$ as a prime with $q = 2^8 \times 13 + 1$ and $n = 256$ as power of two. Kyber is available in several security levels Kyber-512, Kyber-768 and Kyber-1024, which correspond to AES-128, AES-192 and AES-256, respectively. However, the q and n parameters remain constant across all variants. This is one of the other strengths of the Kyber cryptosystem. Kyber's security scaling parameter is $k \in \{2, 3, 4\}$. And Kyber uses other various parameters: η_1, η_2, d_u, d_v, all determined by the security level (k) (see Table 1).

The Kyber-KEM decapsulation process may fail even when all correct input is provided. This failure, known as decapsulation failure, occurs when the shared secret recovered after the decapsulation process is not the same as the shared secret produced during encapsulation. The probability of failure is low but still exists and can be estimated (δ in Table 1).

Basically, the chosen-ciphertext attack (CCA)-secure Kyber-KEM is constructed in two stages. Firstly, establish an indistinguishability under chosen-plaintext attack (IND-CPA) public key encryption scheme, the Kyber.CPAPKE. Secondly, use a variant of the Fujisaki-Okamoto transform [5] to achieve a CCA-secure KEM [3].

2.2.1 Kyber.CPAPKE

Kyber.CPAPKE [3] is a post-quantum public-key encryption scheme using a Module-LWE instance. These algorithms are reused in the different Kyber-KEM algorithms. Kyber.CPAPKE uses three routines: key-generation (Algorithm 1), encryption (Algorithm 2), and decryption (Algorithm 3).

It is relevant to note that the matrix is sampled directly in the NTT domain from a public seed ρ. In contrast, vectors $\mathbf{s}$, $\mathbf{e}$, $\mathbf{r}$, $\mathbf{e_1}$ and the polynomial e_2 are sampled in the coefficient domain in order to ensure good properties (small coefficients). The sample is obtained using the XOF and PRF functions, respectively **SHAKE128** and **SHAKE256**. The Parse function collaborates with XOF to generate uniformly pseudo-random elements of $\mathcal{T}_q$. The CBD_{η_i} function collaborates with PRF to generate samples from the distribution $\mathcal{D}_\eta(\mathcal{R}_q)$.

The Encode algorithm serializes an array of d-bit integers into a $d \times 32$-bytes. The Decode algorithm reverse this process by converting a $d \times 32$-bytes into an array of integers of size d-bit. Both arrays have a fixed length of 256.

Algorithm 1 KYBER.CPAPKE.KeyGen(): Key generation

Output: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := \mathrm{G}(d)$
$N := 0$
for i from 0 to $k-1$ **do**
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathrm{Parse}(\mathrm{XOF}(\rho, j, i))$
  end for
end for
for i from 0 to $k-1$ **do**
  $\mathbf{s}[i] := \mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\sigma, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do**
  $\mathbf{e}[i] := \mathrm{CBD}_{\eta_1}(\mathrm{PRF}(\sigma, N))$
  $N := N + 1$
end for
$\hat{\mathbf{s}} := \mathrm{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathrm{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
$pk := (\mathrm{Encode}_{12}(\hat{\mathbf{t}} \bmod q) || \rho)$
$sk := \mathrm{Encode}_{12}(\hat{\mathbf{s}} \bmod q)$
return (pk, sk)

Algorithm 2 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Input: message $m \in \mathcal{B}^{32}$
Input: random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

$N := 0$
$\hat{\mathbf{t}} := \mathrm{Decode}_{12}(pk)$
$\rho := pk + 12 \cdot k \cdot n/8$
for i from 0 to $k-1$ **do**
  for j from 0 to $k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathrm{Parse}(\mathrm{XOF}(\rho, i, j))$
  end for
end for
for i from 0 to $k-1$ **do**
  $\mathbf{r}[i] := \mathrm{CBD}_{\eta_1}(\mathrm{PRF}(r, N))$
  $N := N + 1$
end for
for i from 0 to $k-1$ **do**
  $\mathbf{e}_1[i] := \mathrm{CBD}_{\eta_2}(\mathrm{PRF}(r, N))$
  $N := N + 1$
end for
$e_2 := \mathrm{CBD}_{\eta_2}(\mathrm{PRF}(r, N))$
$\hat{\mathbf{r}} := \mathrm{NTT}(\mathbf{r})$
$\mathbf{u} := \mathrm{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$
$v := \mathrm{NTT}^{-1}(\hat{\mathbf{t}}^T \circ \hat{\mathbf{r}}) + e_2 + \mathrm{Decompress}_{\mathrm{q}}(\mathrm{Decode}_1(m), 1)$
$c_1 := \mathrm{Encode}_{\mathrm{d_u}}(\mathrm{Compress}_{\mathrm{q}}(\mathbf{u}, d_u))$
$c_2 := \mathrm{Encode}_{\mathrm{d_v}}(\mathrm{Compress}_{\mathrm{q}}(v, d_v))$
return $c := (c_1 || c_2)$

Algorithm 3 KYBER.CPAPKE.Dec(sk, c): decryption

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

1: $\mathbf{u} := \text{Decompress}_{\text{q}}(\text{Decode}_{\text{d}_\text{u}}(\text{c}), d_u)$
2: $v := \text{Decompress}_{\text{q}}(\text{Decode}_{\text{d}_\text{v}}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \text{Decode}_{12}(sk)$
4: $m := \text{Encode}_1(\text{Compress}_{\text{q}}(v - \text{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \text{NTT}(u)), 1))$
5: **return** m

The Compress function takes an element of $\mathbb{Z}_q$ and reduces it into an element of $\mathbb{Z}_{2^d}$ with the function $x \mapsto \lceil (2^d/q) \times x \rfloor$ for $d < 12$. The Decompress function takes an element of $\mathbb{Z}_{2^d}$ and transforms it into an element of $\mathbb{Z}_q$ with the function $y \mapsto \lceil (q/2^d) \times y \rfloor$ for $d < 12$. Two interesting properties are directly obtained:

1. The input is preserved when the Decompress function is follow by the Compress function ($\text{Compress}_d \circ \text{Decompress}_d = Id$). Please note that the inverse composition does not preserve the input, because Compress erases the least significant bits.
2. The input is not significantly distorted by the composition $\text{Decompress}_d \circ \text{Compress}_d$ when d is close to 12.

To improve the efficiency of the Kyber algorithm, the multiplication on $\mathcal{R}_q$ is not performed, instead the Number-Theoretic Transform (NTT) is performed and multiplications are done in the NTT-domain, $\mathcal{T}_q$. It is important to note that $\mathcal{R}_q$ is naturally isomorphic to $\mathcal{T}_q$ through the NTT-transform. So, for $f, g \in \mathcal{R}_q$, we denote $\hat{f} = \text{NTT}(f)$ and we get $f \times_{\mathcal{R}_q} g = \text{NTT}^{-1}(\hat{f} \times_{\mathcal{T}_q} \hat{g})$, where $\times_{\mathcal{R}_q}$ and $\times_{\mathcal{T}_q}$ are multiplications in $\mathcal{R}_q$ and $\mathcal{T}_q$.

2.2.2 FIPS 203: ML-KEM

This section concludes with a comparison of Kyber-KEM and the FIPS-203 standardization (draft). Kyber employs a variant of the Fujisaki-Okamoto (FO) transform, to transition from an IND-CPA encryption scheme to a CCA-secure KEM. The FO-transform version was modified during both round 1 and round 2 of the NIST standardization process, as well as during CRYSTALS-Kyber's standardization period.

Algorithm 4 KYBER.Encaps

Input : Public key pk
Output : Ciphertext c
Output : Shared secret K

1: $m \leftarrow \mathcal{B}^{32}$
2: $m \leftarrow \mathsf{H}(m)$
3: $(\bar{K}, r) := \mathsf{G}(m \parallel \mathsf{H}(pk))$
4: $c := \mathsf{KYBER.CPAPKE.Enc}(pk, m, r)$
5: $K := \mathsf{KDF}(\bar{K} \parallel \mathsf{H}(c))$
6: **return** (c, K)

Algorithm 5 ML-KEM.Encaps

Input : Public key pk
Output : Ciphertext c
Output : Shared secret K

1: $m \leftarrow \mathcal{B}^{32}$
2: ~~$m \leftarrow \mathsf{H}(m)$~~
3: $(K, r) := \mathsf{G}(m \parallel \mathsf{H}(pk))$
4: $c := \mathsf{KYBER.CPAPKE.Enc}(pk, m, r)$
5: ~~$K := \mathsf{KDF}(\bar{K} \parallel \mathsf{H}(c))$~~
6: **return** (K, c)

Algorithm 6 KYBER.Decaps

Input : Secret key sk
Input : Ciphertext c
Output : Shared secret K

$pk \leftarrow sk$
$h \leftarrow sk$
$z \leftarrow sk$
$m' := \mathsf{KYBER.CPAPKE.Dec}(sk, c)$
$(\bar{K}', r') := \mathsf{G}(m' \parallel h)$
$c' := \mathsf{KYBER.CPAPKE.Enc}(pk, m', r')$
if $c = c'$ **then**
    $K := \mathsf{KDF}(\bar{K}' \parallel \mathsf{H}(c))$
else
    $K := \mathsf{KDF}(z \parallel \mathsf{H}(c))$
end if
return K

Algorithm 7 ML-KEM.Decaps

Input : Secret key sk
Input : Ciphertext c
Output : Shared secret K

$pk \leftarrow sk$
$h \leftarrow sk$
$z \leftarrow sk$
$m' := \mathsf{KYBER.CPAPKE.Dec}(sk, c)$
$(K', r') := \mathsf{G}(m' \parallel h)$
$\bar{K} \leftarrow \mathsf{J}(z \parallel c, 32)$
$c' := \mathsf{KYBER.CPAPKE.Enc}(pk, m', r')$
if $c = c'$ **then**
    $K' := \bar{K}$
end if
return K'

Furthermore, NIST introduces four additional modifications between CRYSTALS-Kyber and the standardized version :

- In FIPS-203, CRYSTALS-Kyber (renamed ML-KEM) does not utilize a key derivation function (KDF) on a message hash during the encapsulation process. The ML-KEM and CRYSTALS-Kyber used different building methods for their shared secret. A KDF function was not used to rebuild the shared secret during the decapsulation step. NIST chose to use the shared secret as the first part of $\mathbf{SHA3\text{-}512}(m' \parallel h)$ if the decapsulation was successful. And if the decapsulation fails, a deterministic derivation of a random value is returned (the hash of a random nonce z and the ciphertext c, $\mathbf{SHAKE256}(z \parallel c, 32)$).
- NIST has chosen to set the size of a shared secret at 256 bits.
- In the FIPS-203 version, pre-hashing the message m is no longer necessary, if the message m adheres to the NIST-approved randomness property.
- The final requirement is an input validation step for the ML-KEM.Encaps algorithm. The input for ML-KEM.Encaps(pk) must be a valid encapsulation key.

3 Optimizations

The previous section presented Kyber algorithm. Kyber implementation will be shown in this section. Our considerations are based on the following recommendations [14].

- The crypto implementation must be in constant-time as it is the case with the Kyber reference implementation[2].
- The intermediate values must be destroyed after being used .
- The Kyber reference implementation and the Kyber algorithm can be modified, but they must produce the correct output for every input.

3.1 Algorithm

For each Kyber algorithms (KeyGen, Enc, Dec), the variable dependency is identified. For example in the KeyGen process, where the vector $\mathbf{t}$ is completely determined by three

[2]https://github.com/pq-crystals/Kyber/tree/main

elements, the matrix $\mathbf{A}$, the vector $\mathbf{s}$ and the vector $\mathbf{e}$. Similarly, the matrix $\mathbf{A}$ is totally determined by the value of ρ. In this way, we get a natural hierarchy for each element of Kyber algorithm. If several elements don't have any common dependency, they can be processed sequentially. This identification has two purposes.

- Firstly, the elements for intermediate computations can be destroyed just after being used if they are not needed for further use.
- Secondly, it is also useful for sequencing the different stages of Kyber algorithms.

As an example, in the encryption process of Kyber, it is recommended to finish all computations for c_1 before starting computations for c_2. This process generates fewer variables at the same time. Algorithm 8 illustrates our application of this method to a Kyber key generation process. Our Kyber key generation algorithm (Algorithm 8) enables the manipulation of vector $\mathbf{e}$ after clearing the memory used for matrix $\mathbf{A}$ and vector $\mathbf{s}$. This method can be applied to each algorithm of Kyber.

Algorithm 8 Kyber.CPAPKE.KeyGen(): key generation

Input : Secret key sk
Output : Public key pk

1: $seed \leftarrow \mathcal{B}^{32}$
2: $(\rho, \sigma) := G(seed)$
3: CLEAN d
4: $\hat{\mathbf{A}} := \mathbf{Sample}_{\mathcal{T}_q^{k \times k}}(\rho)$
5: $\mathbf{s} := \mathbf{Sample}_{\mathcal{R}_q^k}(\sigma)_{\mathbf{N}}$
6: $\hat{\mathbf{t}}' := \hat{\mathbf{A}} \circ \mathsf{NTT}(\mathbf{s})$
7: $sk \blacktriangleleft \text{pack}(\hat{\mathbf{s}})$
8: CLEAN $\hat{\mathbf{s}}$, CLEAN $\hat{\mathbf{A}}$
9: $\mathbf{e} := \mathbf{Sample}_{\mathcal{R}_q^k}(\sigma)_{\mathbf{N}}$
10: CLEAN σ
11: $\hat{\mathbf{t}} := \hat{\mathbf{t}}' + \mathsf{NTT}(\mathbf{e})$
12: $pk \blacktriangleleft \text{pack}(\hat{\mathbf{t}}) \| \rho$

3.2 Memory reduction

There are several methods for reducing memory usage. The algorithmic rework method previously exposed in Section 3.1 contributes to the process. In the case of CRYSTALS-Kyber, a streaming method is used to further reduce the memory footprint. This method was introduced in [4], with an implementation of CRYSTALS-Kyber on Cortex-M4 platform and optimization of the NTT with the Kyber's Round 2 specifications.

The streaming method aims to process large elements (matrices and vectors) one polynomial at a time. To generate the complete matrix $\hat{\mathbf{A}}$, k^2 times the Kyber's polynomial memory size is required, and to generate a complete vector, k times the Kyber's polynomial memory size is also necessary. In CRYSTALS-Kyber, each polynomial has coefficients encoded on at most 16 bits, and all coefficients are modulo q ($3329_{10} = 110100000001_2$). Each polynomial has at most 256 coefficients. So, each Kyber polynomial costs 512 bytes of memory. We can estimate the instant memory cost gain to be $(k-1) \cdot 512 + (k \cdot k - 1) \cdot 512$ bytes for a matrix-vector multiplication. The estimated instant memory cost gain is instead $2 \cdot (k-1) \cdot 512$ for the vector-vector multiplication. The equation $\hat{\mathbf{t}}' = \hat{\mathbf{A}} \circ \hat{\mathbf{s}}$ can be solved with this method as shown in the Algorithm 9. This method can also be applied to the equation $\hat{v}' = \hat{\mathbf{t}}^\top \circ \hat{\mathbf{s}}$.

Algorithm 9 streaming $\hat{\mathbf{A}} \circ \hat{\mathbf{s}}$

Input : ρ, σ
Output : $\hat{\mathbf{t}}' = (\hat{t}_0, \hat{t}_1, ..., \hat{t}_{k-1})$

for i from 0 to $k-1$ **do**
  $t_i := 0$
  for j from 0 to $k-1$ **do**
    $s_j \leftarrow Sample_{\mathcal{R}_q}(\sigma)_N$
    $\hat{a}_{ij} \leftarrow Sample_{\mathcal{T}_q}(\rho)$
    $\hat{s}_j := \mathsf{NTT}(s_j)$
    $\hat{t}_i := \hat{t}_i + \hat{a}_{ij} \times_{\mathcal{T}_q} \hat{s}_j$
  end for
  $process(\hat{t}_i)$ $process(x)$ can be any operation using x
end for

When performing matrix-vector multiplication using the streaming method, there is a significant decrease in speed performance. The vector $\mathbf{s}$ is sampled k times and transformed k times in the NTT-form. For CRYSTALS-Kyber, this trade-off between memory and performance is unattractive. Kyber's polynomial weight is 512 bytes, meaning that one polynomial can be preserved in a constrained device memory. To maintain performance, it is recommended to generate the complete Kyber vector prior to matrix-vector multiplication ($\mathbf{s}$ in KeyGen, $\mathbf{r}$ in Enc). However, this slowdown does not happen in the vector-vector multiplication case.

On the fly ("on fly version"). To improve reduction, another method can be applied to Kyber's elements. These elements can be simultaneously sampled and operated upon. Each time enough coefficients have been sampled, they are used directly. This method can be applied to the multiplication operations such as matrix-vector or vector-vector, as well as the addition operations, such as vector-vector or polynomial-polynomial. The addition of noise involving an element with specific properties from $\mathcal{R}_q$ is a specific case. Therefore, elements cannot be directly sampled into $\mathcal{T}_q$ and used. Instead, coefficients are sampled in $\mathcal{R}_q$ and the other operands are re-transformed into a $\mathcal{R}_q$ domain. This process may introduce a slight slowdown depending on the performance of the NTT process. Methods to reduce this slowdown will be discussed in Section 5.

Vector conservation ("store version"). This second version of Kyber's implementation prioritizes reducing memory size, without regenerating the same elements multiple times. Our goal is to provide an implementation with a very small memory footprint that can represent a potential trade-off between memory and performance for extremely constrained devices. As mentioned previously, matrix-vector operations use the operand vectors k times. Re-sampling the same vector represents a significant performance penalty. To avoid generating the same vectors multiple times, we decided to keep them encoded in this implementation version (Fig 1). In Kyber.KeyGen, the vector $\mathbf{s}$ is naturally stored in sk. During the matrix-vector operation, the coefficients of s_j are extracted and multiplied with $\hat{a}_{ij}$ elements. In Kyber.Enc, the vector $\mathbf{r}$ is stored compressed in memory. During $\hat{\mathbf{A}}^{\mathbf{T}} \circ \hat{\mathbf{r}}$, the coefficients of r_j are uncompressed, transform into $\mathcal{T}_q$, and operated on the fly with $\hat{\mathbf{A}}^{\mathbf{T}}$ elements. Again, this method introduces a slowdown. This slowdown is related to the performance of the NTT process. This performance slowdown is presented in Section 4.

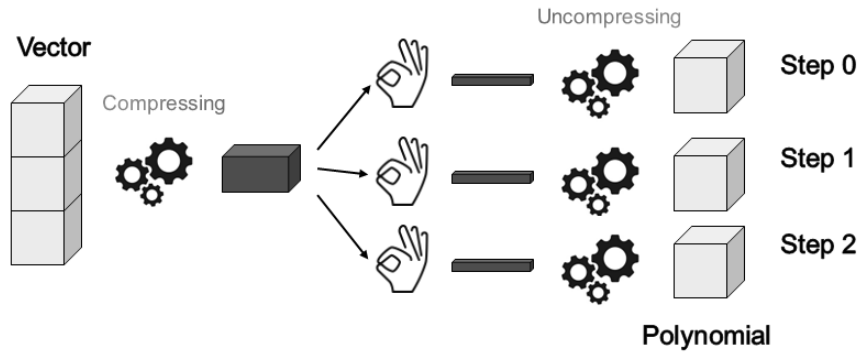


Figure 1: Store mechanism: The vector is compressed and stored in memory. During the calculation step, the relevant polynomial is decompressed and utilized.

3.3 One code, three Kybers

In our work, we merged all variants of CRYSTALS-Kyber for all parameter sets. By merging all variants of CRYSTALS-Kyber, the following advantages can be obtained.

- A better development pipeline and a better testing process.
- A smaller code size.
- Users can select the variant at execution time, as in typical cryptographic libraries.

All three versions of Kyber are compiled into a single code using the method of code factorization. Code factorization improves readability. It also allows you to work on all three variants simultaneously. In addition, testing of these three variants can be done in a single compilation. While code size may not be a significant issue for larger devices, it is crucial for embedded systems. In an embedded environment, small code size is a great advantage, as it allows for more functionality to be carried on the same micro-controller. In our code, we pre-build some data structures to modify the behavior of our program. At the time of executing the Kyber variant, we loaded one of these structures into memory to provide context. This merge comes with a cost of instant memory footprint at a particular step of Kyber's algorithm. However, this instant memory footprint cost is not at the peak memory consumption moment: this cost is thus transparent for the maximum memory footprint.

4 Results

We implemented CRYSTALS-Kyber using the methods described in Section 3. Our work aims to reduce the memory footprint without significantly slowing down the implementation's performance when compared to the reference implementation. To achieve this, we avoided over-commitment and maintained stable performance by generating specific vectors. Additionally, we implemented a "store" version of the Kyber crypto-system. The "store" implementation presents a significant reduction in memory usage compared to other implementations. However, Kyber.Encaps("store") and Kyber.Decaps("store") experience a significant slowdown of more than 22% compared to the reference implementation. On the other hand, Kyber.Keygen("store")' s memory usage has been drastically reduced with a slowdown of less than 10% (4.4% for Kyber-512 and 7.4% for Kyber-768). We have updated both of our implementations in order to meet the requirements of ML-KEM.

Table 2: Stack usage for all three security levels of Kyber and ML-KEM update.

scheme	implementation	Keygen	Encaps	Decaps
Kyber512	reference*	6 116 B	8 780 B	9 556 B
Kyber512	[4] Botros et al.*	3 136 B	2 720 B	2 744 B
Kyber512	this work("on fly")*	2 576 B (-17%)	2 680 B (-1%)	2 680 B (-2%)
Kyber512	this work("store")*	2 088 B (-33%)	2 440 B (-10%)	2 440 B (-11%)
ML-KEM512	this work("on fly")*	2 576 B (-17%)	2 680 B (-1%)	2 680 B (-2%)
ML-KEM512	this work("store")*	2 088 B (-33%)	2 440 B (-10%)	2 440 B (-11%)
Kyber768	reference*	10 212 B	13 380 B	14 476 B
Kyber768	[4] Botros et al.*	3 648 B	3 232 B	3 248 B
Kyber768	this work("on fly")*	3 088 B (-15%)	3 192 B (-1%)	3 192 B (-2%)
Kyber768	this work("store")*	2 088 B (-43%)	2 568 B (-20%)	2 568 B (-21%)
ML-KEM768	this work("on fly")*	3 088 B (-15%)	3 192 B (-1%)	3 192 B (-2%)
ML-KEM768	this work("store")*	2 088 B (-43%)	2 568 B (-20%)	2 568 B (-21%)
Kyber1024	reference*	15 100 B	18 772 B	20 348 B
Kyber1024	[4] Botros et al.*	4 160 B	3 752 B	3 776 B
Kyber1024	this work("on fly")*	3 600 B (-13%)	3 704 B (-1%)	3 704 B (-2%)
Kyber1024	this work("store")*	2 088 B (-50%)	2 696 B (-28%)	2 696 B (-29%)
ML-KEM1024	this work("on fly")*	3 600 B (-13%)	3 704 B (-1%)	3 704 B (-2%)
ML-KEM1024	this work("store")*	2 088 B (-50%)	2 696 B (-28%)	2 696 B (-29%)

* The reference FIPS-202 operation requires 440 bytes of memory.

There is no reference implementation of ML-KEM to which to refer. All the different memory and speed measurements are reported in Tables 2 and 3.

One advantage of our implementation is that it can be deployed in standard products that are already in the field. The implementation was developed in embedded C to ensure cross-platform compatibility. Our implementation does not use any architecture-specific method. We have tested our implementation on several embedded platforms. Memory performances are reported in Table 2 (in bytes), and speed performances are reported in Table 3 (in cycle on SC300 CPU with systick timer). [1, 4, 6, 9, 15, 16] present similar work but with different approaches, especially in terms of speed performance. Some of these studies employ architectural features of their platform or implement specific cryptographic primitives in order to optimize memory usage. Consequently, the Kyber reference implementation is employed as a reference for speed performance.

Our implementation integrates all variants of Kyber's security level simultaneously. The total code size is reduced by a factor of three as a result of this code merge. This limits the area of code to be uploaded to devices with reduced non-volatile memory. The security variant is selected at runtime as software cryptographic libraries.

The objective of these optimisations is to facilitate the compatibility of post-quantum cryptography with microcontrollers utilised in the Internet of Things. In this context, devices are constrained in terms of memory, with both volatile and non-volatile storage being limited. For all reference measurements, the platform utilises an ARM SC 300 core, 24KB of RAM memory and operates at 20MHz. All code is compiled in high memory optimisation with the IAR compiler.

Table 3: Cycle for all three security levels of Kyber and ML-KEM update.

scheme	implementation	Keygen	Encaps	Decaps
Kyber512	reference	1 621 975 c	2 082 797 c	2 021 456 c
Kyber512	this work("on fly")	1 653 066 c (+2%)	1 978 020 c (-5%)	1 915 921 c (-5%)
Kyber512	this work("store")	1 693 116 c (+4%)	2 548 365 c (+22%)	2 504 448 c (+24%)
ML-KEM512	this work("on fly")	1 653 066 c (+2%)	1 652 747 c (-26%)	1 873 463 c (-8%)
ML-KEM512	this work("store")	1 693 116 c (+4%)	2 223 092 c (+6%)	2 465 373 c (+18%)
Kyber768	reference	2 592 513 c	3 299 665 c	3 196 016 c
Kyber768	this work("on fly")	2 693 535 c (+4%)	3 193 031 c (-3%)	3 085 676 c (-3%)
Kyber768	this work("store")	2 783 764 c (+7%)	4 479 157 c (+36%)	4 411 754 c (+38%)
ML-KEM768	this work("on fly")	2 693 535 c (+4%)	2 747 694 c (-20%)	3 041 541 c (-5%)
ML-KEM768	this work("store")	2 783 764 c (+7%)	4 033 820 c (+18%)	4 372 691 c (+26%)
Kyber1024	reference	no available	no available	no available
Kyber1024	this work("on fly")	4 215 270 c	4 813 810 c	4 703 649 c
Kyber1024	this work("store")	4 373 760 c	7 100 965 c	7 021 106 c
ML-KEM1024	this work("on fly")	4 215 270 c	4 243 009 c	4 617 677 c
ML-KEM1024	this work("store")	4 373 760 c	6 530 264 c	6 981 895 c

5 Conclusions and Future Works

This paper presents several optimization techniques for memory efficiently implementing Kyber-KEM. The techniques can be deployed in products already in the field, and result in a significant reduction in Kyber memory usage. Specifically, the memory reduction represents 17% for Kyber.KeyGen, while Kyber.Enc and Kyber.Dec results are similar. The "store" implementation represents an option for hardly constrained devices as in IoT context, but presents some slowdowns. The KeyGen function offers a significant memory reduction (30% to 50%) without significantly reducing the speed performance. These optimization techniques are suitable for constrained environments, such as microcontrollers and smart card devices.

Afterwards several methods for improving CRYSTALS-Kyber speed performance are compared, with references to relevant articles.

Firstly, we would like to present the execution time of each different activity during the Kyber's algorithm. Kyber execution time is shared between hashing, NTT-transformation, modular reduction, coefficient-wise multiplication, bit-packing and other computations (Fig 2). With this representation, we can select the most suitable part for improving Kyber's implementation performance.

If performance is a primary challenge, a Kyber dedicated hardware intellectual property (I.P.) may be a viable solution [11, 17]. But, this would require, for a chip manufactures, redesigning the platforms to integrate a Kyber hardware I.P. As previously said, a benefit of our solution is that can be integrated in devices in the field. It is important to note that the most time-consuming operation is the hashing process (Fig 2). Therefore, a Keccak hardware I.P. appears to be a suitable compromise between development, integration, surface cost, and performance gain. The performance and architecture of the Keccak hardware IP are described in [2] by Assad et al.

Another method to enhance Kyber's efficiency is to accelerate the NTT-transformation process. The most common approach is to take advantage of the specific platform architecture. The benefits of this optimization technique are platform-dependent. In [4], Botros et al. have presented an optimization technique for the NTT-transform in Kyber on the Cortex-M4 platform. They point to several factors for improvement, many of which

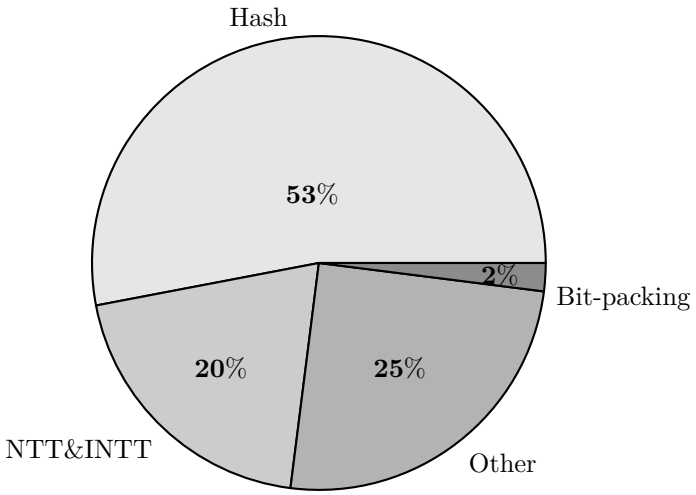


Figure 2: Kyber operation breakdown

are based on assembly language instructions. In [6] Greconici presents an optimization of Kyber's NTT at assembly language level on the RISC-V architecture. As in [4], the author utilizes the instruction set and architecture to improve execution speed. [4] and [6] have both studied the CRYSTALS-Kyber round 2 specifications on a 32-bit architecture. Additionally, [15] and [9] have presented several optimization techniques on CRYSTALS-Kyber for 64-bit architecture. A 64-bit ARM Cortex-A processor architecture was utilized to enhance the efficiency of both the NTT-transform and modular reduction.

Some of these approaches can be combined with all our optimizations. In future work, we will continue to understand how to improve the performance reduction of a masked version of CRYSTALS-Kyber implementation and whether it introduces leaks.

References

[1] Amin Abdulrahman, Vincent Hwang, Matthias J. Kannwischer, and Amber Sprenkels. Faster kyber and dilithium on the cortex-m4. Cryptology ePrint Archive, Paper 2022/112, 2022.

[2] F. Assad, F. Elotmani, M. Fettach, and A. Tragha. An optimal hardware implementation of the keccak hash function on virtex-5 fpga. https://ieeexplore.ieee.org/document/9028020, 2019.

[3] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, , and Damien Stehlé. Crystals-kyber (version 3.02). https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf, 2020.

[4] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. Memory-efficient high-speed implementation of kyber on cortex-m4. https://eprint.iacr.org/2019/489, 2019.

[5] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings, volume 1666 of Lecture Notes in Computer Science, pages 537–554. Springer, 1999.

[6] Denisa GRECONICI. KYBER on RISC-V. PhD thesis, Radboud University Nijmegen, Nijmegen, Gelderland, Netherlands, January 2020.

[7] IBM. Ibm unveils 400 qubit-plus quantum processor and next-generation ibm quantum system two. https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two, 2022.

[8] Adeline Langlois and Damien Stehle. Worst-case to average-case reductions for module lattices. https://eprint.iacr.org/2012/090, 2012.

[9] Zhao Lirui, Zhang Jipeng, Huang Junhao, Liu Zhe, and Hancke Gerhard. Efficient implementation of kyber on mobile devices. https://ieeexplore.ieee.org/document/9763781, 2021.

[10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. https://eprint.iacr.org/2012/230.pdf, 2010.

[11] Ziying Ni, Ayesha Khalid, Dur e Shahwar Kundi, Máire O'Neill, and Weiqiang Liu. Hpka: A high-performance crystals-kyber accelerator exploring efficient pipelining. https://eprint.iacr.org/2022/1093, 2022.

[12] NIST. Report on post-quantum cryptography. https://doi.org/10.6028/NIST.IR.8105, 2016.

[13] NIST. Status report on the third round of the nist post-quantum cryptography standardization process. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf, 2022.

[14] NIST. Fips-203 (draft) module-lattice-based key-encapsulation mechanism standard. https://csrc.nist.gov/pubs/fips/203/ipd, 2023.

[15] Pakize Sanal, Emrah Karagoz, Hwajeong Seo, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. Kyber on arm64: Compact implementations of kyber on 64-bit arm cortex-a processors. https://eprint.iacr.org/2021/561, 2021.

[16] Seog Chung Seo and HeeSeok Kim. Portable and efficient implementation of crystals-kyber based on webassembly. https://www.techscience.com/csse/v46n2/51636/html, 2023.

[17] Guo Wenbo, Li Shuguo, and Kong Liang. An efficient implementation of kyber. https://ieeexplore.ieee.org/document/9509281, 2022.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 25–31.

Another Version of Chosen Plaintext Attack on McEliece Cryptosystem

Vaishnavi Nagaraja[1], Terry Shue Chien Lau[2] and Muhammad Rezal Kamel Ariffin[3]

[1] Institute for Mathematical Research, Universiti Putra Malaysia
gs62737@student.upm.edu.my
[2] Faculty of Computing and Informatics, Multimedia University
terry.lau@mmu.edu.my
[3] Department of Mathematics and Statistics, Universiti Putra Malaysia
rezal@upm.edu.my

Abstract. McEliece Cryptosystem is one of the well-known cryptosystems in code-based cryptography. It is also one of the post-quantum cryptosystems. The code-based cryptosystem relies its security on the hardness of the syndrome decoding problem (SDP) when the code is random. There are a lot of studies have been carried out to determine the message or plaintext without the need to solve the hard problem, SDP. One of the well-known alternative or non-structural attacks to retrieve the messages is the information set decoding attack (ISD). In this paper, we would like to propose a generic decoding algorithm based on the motivation of Prange's algorithm to retrieve the plaintext partially or successfully.

Keywords: McEliece Cryptosystem · Plaintext Attack

1 Introduction

Cryptography is a secure communication system where it utilizes the mathematical concepts and algorithms to transform messages or plaintext in ways so that it is hard to retrieve back the message. There are many well-known cryptosystems in the current era, such as RSA and Elliptic Curve Cryptography (ECC). However, these cryptosystems suffer from some weaknesses that might lead to the vulnerability of attacks such as the Wiener attack on RSA [5]. The hard problem of RSA which is the factorization of large prime numbers could turn out to be its weakness if there exists a quantum computer in the future. One of the easiest solutions can be done to overcome this issue is by implementing post-quantum cryptography in the current technology. Post-quantum cryptography is the cryptosystem that is resistant to current computer's and quantum computer's attacks in the future. There are few post-quantum cryptosystems such as multivariate, lattice-based and code-based. One of the most common candidates for post-quantum cryptosystems is built based on code-based cryptography such as HQC and BIKE. McEliece cryptosystem [2] is one of the most well-known and the first motivation initiated in code-based cryptosystems almost 40 years ago which utilizes the Goppa code (algebraic coding theory). McEliece is one of the fourth-round candidates of KEMs algorithm standards of the National Institute of Standards and Technology (NIST). The hard problem for McEliece is known as syndrome decoding problem (SDP).

The ciphertext, c equation of McEliece cryptosystem is given as $c = mG + e$ where $m \in F_q^k$ is the message, $G \in F_q^{k \times n}$ is a generator matrix and $e \in F_q^n$ is the error vector.

We need to apply the correct decoding algorithm (solves the hard problem) to remove the error from the ciphertext and retrieve the message correctly. McEliece is one of the secure cryptosystems since there are no well-known attacks have been performed on Goppa code of McEliece since 1978. It is proven to be resistant to quantum attacks. Due to the construction of the difficult code, attacks are difficult to be perform on it. Therefore, the security of McEliece is tight and researchers are working on attacks that can be performed. The earliest proposal that can be worked on McEliece is the generic decoding algorithm attacks. Information set decoding (ISD) attack [1] is one of the well-known attack algorithms for decoding a general linear code. This attack is efficient under the hamming metric and there are many improvised versions of it such as Lee-Brickell (1988) [3], Leon (1988) [4], and Bernstein et.al (2011) [6]. Research has been done to improve the previous way of performing information set decoding attacks that can have a better complexity. Based on the motivation of ISD attack, we would like to propose a simple version of another generic decoding attack. We utilize the attack on McEliece's ciphertext pattern. This proposed attack seems to be a much simpler implementation compared to ISD attack.

1.1 Research Flow

In this paper, we provide some preliminaries that are going to be used in our paper to propose these generic decoding attack. Then, we provide the general generic decoding algorithm proposed by Prange (1962) [1]. We also propose a new generic decoding algorithm by providing an alternative way to perform the generic decoding algorithm other than the information set decoding (ISD) attack.

2 Preliminaries

In this section, we will look into the hamming metric and the hard problem which is known as the syndrome decoding problem (SDP). Then, we explain briefly the information set decoding attack.

Definition 1 (Linear Code)**.** A *linear code* $\mathcal{C}$ is a subspace of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is a finite field with q elements of length n. If $\mathcal{C}$ has dimension k, it is referred to as an $[n, k]$-linear code. Elements of $\mathcal{C}$ are called *codewords*.

Definition 2 (Hamming Distance)**.** The *Hamming distance* $d(x, y)$ between two codewords, $x, y \in \mathbb{F}_q^n$ is defined as the number of positions where x and y differ. The *Hamming weight*, $wt(x)$ of a codeword, x is the number of non-zero entries in x. The *minimum distance*, $d_{\min}$ of the code, $\mathcal{C}$ is the smallest hamming distance between any two distinct codewords in $\mathcal{C}$.

Definition 3. A matrix $G \in F_q^{k\times n}$ is called a generator matrix of code C if its rows form a basis of C. A matrix H is called a parity check matrix of C if $C = \{x \in F_q^n : H \cdot x^T = 0\}$.

Definition 4 (Syndrome Decoding (SD) Problem)**.** Given a random matrix $H_q^{(n-k)\times n}$, a random vector, $s \in F_q^{(n-k)}$ and an integer $w > 0$ as an input. The syndrome decoding problem (q, n, k, w) needs to determine a vector $e \in F_q^n$ such that it satisfies $wt_H(e) = w$ and $s = eH^T$.

The well-known McEliece cryptosystem (code-based cryptosystem) is a public-key cryptosystem based on the hardness of syndrome decoding problem under the Hamming metric. It relies on binary Goppa code, a class of error-correcting code. It employs an $[n, k]$-linear code with a secret generator matrix, G. The security of the McEliece cryptosystem relies on the difficulty of decoding a received vector to the nearest codeword in the presence of errors under the Hamming metric.

2.1 McEliece

McEliece is one of the most popular code-based cryptosystems. The following outlines the key generation, encryption and decryption steps involved in the McEliece cryptosystem:

2.1.1 Key Generation

1. Choose an $[n, k]$-linear code C with a generator matrix, G and a parity-check matrix, H.
2. Select a random $k \times k$ invertible matrix, S over $\mathbb{F}_q$.
3. Select a random $n \times n$ permutation matrix, P.
4. Compute the public generator matrix, $G' = SGP$.
5. Publish G' as the public key and keep S, G, and P as the private key.

2.1.2 Encryption

To encrypt a message, $m \in \mathbb{F}_q^k$:

1. Represent the message, m as a vector in $\mathbb{F}_q^k$.
2. Generate a random error vector, $e \in \mathbb{F}_q^n$ with weight, $wt(e) \leq t$, where t is the error-correcting capability of the code.
3. Compute the ciphertext, $c = mG' + e$.

2.1.3 Decryption

To decrypt the ciphertext, c :

1. Compute $c' = cP^{-1}$ to obtain $c' = mSG + eP^{-1}$.
2. Decode c' using the standard decoding algorithm for the code $\mathcal{C}$ to find the original message, mS and the error vector, e'. (This step solves our hard problem and removes the error vector).
3. Compute $m = (mS)S^{-1}$ and obtained the message, m.

The linear code used in the generator matrix of McEliece is the Goppa code. It was proven that the McEliece cryptosystem is vulnerable to the chosen-plaintext attack due to some bad choices of parameters. The attack below is one of the generic decoding algorithms that can be applied to McEliece cryptosystem based on the chosen parameters. An Information Set Decoding (ISD) attack is a class of algorithms that used for decoding linear codes. The general idea behind ISD is to select randomly a subset of positions, called an *information set*, where the submatrix formed by these positions is assumed to be invertible. The algorithm then attempts to solve for the error vector given the observed syndrome. Various improvements and variants of ISD exist, each optimizing different aspects of the algorithm to enhance the efficiency and reduce the complexity.

2.2 ISD attack

We describe the basic general concept of the ISD attack in McEliece. Given SD problem, syndrome, $s = eH^T$, we need to find the error vector, e. When we apply the syndrome to McEliece ciphertext, $c = mG + e$ will become $cH^T = s$.

2.2.1 ISD Algorithm

The task of the ISD algorithm is to recover the vector, m from $c = mG+e$, where $G \in \mathbb{F}_q^{k\times n}$ is a generator matrix of a random $[n, k, d]$-linear code, $\mathcal{C}$ and $\mathrm{wt}_H(e) \leq t = \left\lfloor \frac{d-1}{2} \right\rfloor$. The following are the steps of the ISD algorithm:

1. Randomly choose a set, K of k indices, i.e., $K \subseteq \{1, \ldots, n\}$ with $|K| = k$.

2. Compute $m' = c_K G_K^{-1}$ and $e' = c - m'G$. If $\mathrm{wt}(e') \leq t$, then output m'; else return to Step 1.

As we know, the ciphertext, $c = mG+e$, which implies that we choose $c_K = mG_K + e_K$ for ISD attack. If $e_K = 0_k$, then $c_K = m'G_K$ and $c - m'G = c - c_K G_K^{-1} G = e$. Therefore, $c - m'G$ should have weight t.

2.2.2 Explanation of ISD attack

Step 1: Choosing the Information Set
Randomly select a subset, K of k indices from the set $\{1, \ldots, n\}$. The subset, K is known as the information set.
Step 2: Computing the Estimated Message and Error

- Compute the estimated message, m' using the selected indices: $m' = c_K G_K^{-1}$.

- Calculate the error vector, e' as $e' = c - m'G$.

- If the Hamming weight of e' is lesser than or equal to t, output m' as the recovered information vector.

- If the condition is not met, repeat the process by returning to Step 1.

This iterative process continues until the correct information vector, m is found, ensuring that the error vector, e' has the desired Hamming weight, t.

3 Proposed Generic Decoding Algorithm

The algorithm we propose builds upon the some principles of ISD such as assuming error vector, $e = 0$. Our proposed algorithm also incorporates novel techniques to improve decoding efficiency under the Hamming metric. We introduce a modified selection strategy for the error-correction mechanism. We mainly utilize the number theory concepts such as using modular arithmetic $(mod\ q)$.

Let the ciphertext, c of McEliece as $c = mG + e \in F_q^n$ where G is a generator matrix, m is the plaintext or message, and e is the error vector. Then, we have,

$$c = mG + e$$

$$[c_1, c_2, ..., c_n] = [m_1, m_2, ..., m_k] \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & g_{k3} & \cdots & g_{kn} \end{pmatrix} + [e_1, e_2, ..., e_n]$$

$$[c_1, c_2, ..., c_n] = [m_1, m_2, ..., m_k] \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & g_{k3} & \cdots & g_{kn} \end{pmatrix} + [e_1, e_2, ..., e_n] \pmod q$$

$$[c_1, c_2, ..., c_n] = [m_1, m_2, ..., m_k] \begin{pmatrix} g_{11} & g_{12} & g_{13} & \cdots & g_{1n} \\ g_{21} & g_{22} & g_{23} & \cdots & g_{2n} \\ g_{31} & g_{32} & g_{33} & \cdots & g_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{k1} & g_{k2} & g_{k3} & \cdots & g_{kn} \end{pmatrix} + [e_1, e_2, ..., e_n] - q[z_1, z_2, ..., z_n] \tag{1}$$

Based on the equation (1), we look into our proposed decoding algorithm by looking specifically into the position of ciphertext, c_i where $i = 1, ..., n$ as below:

$$c_i = [m_1, m_2, ..., m_k] \begin{pmatrix} g_{1i} \\ g_{2i} \\ g_{3i} \\ \vdots \\ g_{ki} \end{pmatrix} + e_i - qz_i$$

$$c_i = m_1 g_{1i} + m_2 g_{2i} + m_3 g_{3i} + ... + m_k g_{ki} + e_i - qz_i \tag{2}$$

$$c_i = mG_i + e_i - qz_i \tag{3}$$

Theorem 1 (Decoding Technique)**.** *Let g_{Bi} be the largest element in $\{g_{1i}, g_{2i}, g_{3i}, ..., g_{ki}\}$ and $g_{Bi} \sim 2^B$. Then, we have $2^B \leq mG_i \leq 2^{B+1}$, and $2^{B-1} \leq z_i \leq 2^B$. Assume the elements in $\{g_{1i}.g_{2i}, ..., g_{ki}\}$ are not exponentially large and $e_i = 0$. Then, we can brute force the non-negative values of $z_i \in [2^{B-1}, 2^B]$ to obtain the message, $m'_p \in \mathbb{Z}$ where $p = 1, ..., k$.*

Proof. For each brute force of values z_i, do $Y_p \equiv A \pmod{g_{pi}}$ where $p = 1, 2, ..., k$ and $A = mG_i = c_i + qz_i$. For each element of g_{pi} from $\{g_{1i}.g_{2i}, ..., g_{ki}\}$, do $m'_p = \frac{A - Y_p}{g_{pi}} \pmod q$. After obtaining $m' = m'_p$ for $p = 1, 2, ..., k$, check whether $wt(e) \leq t$ by putting back the message, m' into $e = c - m'G$. □

Using this theorem, we can determine the message, m partially or fully successfully with the correct choices of q-values. If we choose a small q, we can use this generic decoding technique successfully. This method can be done successfully for the assumption of the error vector, $e_i = 0$ using this simple algebra concept. After obtaining the message string fully, m', we can put it back into the $e = c - m'G$ and check whether it satisfies $wt(e) \leq t$.

3.0.1 Proposed Algorithm

The objective of proposed algorithm is to recover the vector, m from $c = mG + e$, where $G \in \mathbb{F}_q^{k \times n}$ is a generator matrix of a random $[n, k, d]$-linear code, $\mathcal{C}$ and $\text{wt}(e) \leq t = \left\lfloor \frac{d-1}{2} \right\rfloor$.

The following are the detailed steps of the proposed algorithm:

1. Randomly look into the ciphertext, $c = (c_1, ..., c_n)$ and choose one of the c_i where $i = 1, 2, ..., n$.
2. Choose the non-negative value of $z_i \in [2^{B-1}, 2^B]$ according to the Theorem 1. Then, compute $A = c_i + qz_i$.
3. Compute $Y_p \equiv A \pmod{g_{pi}}$ where $p = 1, 2, ..., k$.
4. Compute $m'_p = \frac{A-Y_p}{g_{pi}} \pmod{q}$ for each element of g_{pi} from $\{g_{1i}.g_{2i}, ..., g_{ki}\}$ and create the message vector, m'.
5. Repeat same step 1-4 for all possible c_i and z_i where $i = 1, 2, ..., n$.
6. Compare all the message vector, m' we obtained from c_i and choose the similar message pattern, m' as our message, m.
7. Finally, compute $e = c - m'G$ and check whether $wt(e) \leq t$.

4 Conclusion

In this paper, we proposed a new generic decoding algorithm based on the motivation of ISD attack [1]. We used some of the simple principles that had been utilized in the ISD attack. We also used some simple algebra techniques based on the number theory to propose this attack to retrieve the correct message, m. Since this was a novel idea in code-based cryptography, we proposed this first approach that can be extended and improved further in the future. Some problems had been discovered in this proposed generic algorithm such as the proposed theorem can be modified further by providing some conditions so that the theorem could be used in all cases. However, this attack can be implemented into all other code-based cryptosystems for a suitable ciphertext pattern.

Acknowledgments

We would like to express our gratitude to the Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM) in allowing research to be conducted. We also want to thank the MYBRAINSC scholarship scheme from the Ministry of Higher Education of Malaysia. Finally, we appreciate the editor and referees for the comments that can improve this paper.

References

[1] E. Prange, The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory* **1962**, *8(5)*, 5–9.

[2] R.J. McEliece, A public-key cryptosystem based on algebraic. *Coding Thv* **1978**, *4244*, 114–116.

[3] P. J. Lee and E. F. Brickell, An observation on the security of McEliece's public-key cryptosystem. *Theory and Application of Cryptographic Techniques (EUROCRYPT 1988)* **1988**, 275–280.

[4] J.S. Leon, A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, **1988**, *34(5)*, 1354-1359.

[5] M. J. Wiener, Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information theory*, **1990**, 553-558.

[6] D.J. Bernstein, T. Lange & C. Peters, Smaller decoding exponents: ball-collision decoding. *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA*, **2011**, 743-760.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 32–47.

A New Generic Strategy for Solving Multivariate Quadratic Polynomials

Kamilah Abdullah[1,3], Muhammad Rezal Kamel Ariffin[1,2] and Nurul Amiera Sakinah Abdul Jamal[1]

[1] Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia. kamilah@tmsk.uitm.edu.my,rezal@upm.edu.my,amierasakinah@gmail.com
[2] Department of Mathematics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia. rezal@upm.edu.my
[3] School of Mathematical Sciences, College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia. kamilah@tmsk.uitm.edu.my

Abstract.
A Multivariate Public-Key Cryptosystem (MPKC) is a potential candidate for post-quantum cryptography. The security of an MPKC relies on the difficulty of solving systems of multivariate quadratic (MQ) polynomial equations over finite fields. Specifically, the MQ problem, which involves solving such a system of quadratic polynomials, is a significant research area in this field. This study focuses on solving the MQ problem using a solution approach based on the determinant and inverse of the modulo matrix. The process involves verifying the existence of the determinant and the invertibility of the matrix public key, denoted as $\mathcal{P}$. The candidate solutions are obtained by multiplying the inverse of the matrix by a vector, expressed as $\mathbf{x} = \mathcal{P}^{-1}\mathbf{B}$. Additionally, we provide an example from an established MPKC to illustrate our proposed method. In this study, we introduce a novel method for solving MQ problems based on a system of non-linear equations and matrices. The process involves verifying the existence of the determinant and the invertibility of the matrix public key, $\mathcal{P}$. The idea is to apply the method of $\mathcal{P}\mathbf{x} = \mathbf{B}$, where the matrix $\mathcal{P}$ can be either a square or a non-square matrix. The result shows that the possible candidates can be obtained by solving $\mathbf{x} = \mathcal{P}^{-1}\mathbf{B}$.

Keywords: Post-Quantum Cryptography · Multivariate Public-Key Cryptosystem · Multivariate Quadratic Problem · Modulo inverse matrix · Square matrix · Determinant modulo matrix

1 Introduction

Post-quantum cryptography (PQC) concerns cryptographic methods specifically developed to withstand the potential risks presented by quantum computers. Classical cryptographic systems, such as RSA [24] and ECC (Elliptic Curve Cryptography), depend on the computational complexity of such problems as integer factorization and discrete logarithms. Nevertheless, with the emergence of quantum computers with adequate computational capacity, these difficulties can be easily solved by employing algorithms such as Shor's algorithm [25]. Consequently, classical encryption methods become susceptible to compromise. PQC aims to provide cryptographic methods that maintain their security even in the event of widespread adoption of quantum computing. This field concentrates explicitly on alternative mathematical frameworks hypothesised to possess a high level of resilience against quantum attacks.

To address the possible threat posed by quantum computing, organizations like the National Institute of Standards and Technology (NIST) have been actively involved in establishing standardized cryptographic algorithms that can withstand attacks from quantum computers. In 2016, NIST began a multi-round process intending to assess and develop standard quantum-resistant public-key cryptography algorithms. The main goal is to build strong and secure cryptography standards prior to the emergence of widespread quantum computers [1, 2, 23].

The need to develop Post-Quantum Cryptography (PQC) arises from the swift progress in quantum computing. Despite the absence of existing large-scale quantum computers, the potential threat they provide has led to the implementation of preventive actions [4]. Governments, industry, and academics are aggressively pursuing the research and standardization of Post-Quantum Cryptography (PQC) to guarantee data security that remains effective in the face of future advancements in quantum computing.

Critical families of post-quantum cryptographic algorithms include: Lattice-based cryptography utilizes the hardness of lattice problems [21], such as the Shortest Vector Problem (SVP) [18] and Learning With Errors (LWE) [22]. Code-based cryptography is based on the difficulty of decoding random linear codes, with notable examples being the McEliece and Niederreiter cryptosystems [26]. Multivariate public-key cryptography involves solving systems of multivariate quadratic equations, which are problematic for both classical and quantum computers [10]. Hash-based cryptography relies on the security of hash functions, with schemes like the Merkle signature scheme offering strong post-quantum security [5]. Isogeny-based cryptography utilizes the difficulty of finding isogenies between elliptic curves, representing a relatively new area of post-quantum cryptography [9].

Recent research in multivariate public-key cryptosystems (MPKCs) has stimulated the development of new methods for solving multivariate polynomial equations over finite fields, including techniques such as XL [6–8, 27] and Gröbner basis methods F4 and F5 [11, 12]. Many MPKCs utilize quadratic polynomials, making the challenge of solving systems of multivariate quadratic (MQ) polynomial equations, known as the MQ problem, an NP-hard [14, 15]. The security of MPKCs relies on the computational difficulty of solving the MQ problem, making efficient solutions to this problem a crucial area of focus.

Matrix operations become highly relevant when addressing the MQ problem. Matrix operations are essential in mathematics, computer science, and engineering, offering effective tools for addressing intricate problems involving equations, transformations, and related tasks. Specifically, the concept of the matrix inverse plays a critical role, particularly in linear algebra, where it is used to solve systems of linear equations. When these ideas are applied to modular arithmetic, the inverse modulo matrix becomes a crucial structure with numerous applications in cryptography, coding theory, and computational mathematics. By leveraging these matrix operations, researchers can develop more efficient methods for tackling the MQ problem, thereby enhancing the security and performance of MPKCs.

In this paper, we introduce the exploration of solving the MQ problem by transforming the system of MQ polynomial into a system of $\mathcal{P}\mathbf{x} = \mathbf{B}$. Given the system of MQ polynomial in two cases:

1. The system $\mathcal{P}$ is a square $n \times n$ matrix. Then, the system can be solved as $\mathbf{x} = \mathcal{P}^{-1}\mathbf{B}$ to obtain the possible candidate of the solution.

2. The system $\mathcal{P}$ is not a square matrix, where the number of equations is less than monomials or the number of equations exceeds monomials.

   (a) If the number of rows is less than the number of monomials, then additional rows must be added to match the number of monomials. Furthermore, each element in the rows must be neither identical nor proportional to any other row.

(b) If the number of rows exceeds the number of monomials, omit the excess rows to form a square matrix and solve it as in case 1.

1.1 Comparison of Performance Between Our Method and Existing Methods

Table 1: The performance of our method in comparison to existing methods.

Name of method	Structural Dependency	Computational Complexity	Potential Security Implication
Our method	Yes (see section 4)	Polynomial	High if it falls under the defined structure
XL method	No	Polynomial	High if the exact dimension of $\mathcal{I}_D$ is $\geq \mathcal{T} - 1$, see [7].
Gröbner basis	No	Polynomial	High if Buchberger criterion is satisfied, see [12, 13].

Table 1 compares the performance of our approach with existing methods in terms of structural dependency, computational complexity, and potential security implications. The potential security implication refers to the attack on the system of equations.

2 System of Linear Equations and Matrices

In science and mathematics, information is frequently organized in rows and columns to form rectangular arrays known as matrices. These matrices often represent tables of numerical data derived from physical observations but also appear in various mathematical contexts. This is particularly important for developing computer programs designed to solve systems of linear equations, as computers are highly efficient at manipulating arrays of numerical data. Every system of linear equations will either have no solutions, exactly one solution, or infinitely many solutions [3].

2.1 Matrix Form of a Linear System

Matrix multiplication is a crucial tool in solving systems of linear equations. Consider a system of m equations with n unknowns [3].

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m \end{aligned}$$

Given that two matrices are considered equal if and only if their corresponding entries are similar, we can simplify the system of equations by replacing the m equations with a single matrix equation.

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

The left side of this equation can be expressed as the product of a $m \times 1$ matrix.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

If we denote these matrices as $\mathcal{P}$, $\mathbf{x}$, and $\mathbf{B}$ accordingly, the original system of m equations with n unknowns has been exchanged with the established matrix equation $\mathcal{P}\mathbf{x} = \mathbf{B}$. The matrix $\mathcal{P}$ in this equation is referred to as the coefficient matrix of the system. The equation $\mathcal{P}\mathbf{x} = \mathbf{B}$, where $\mathcal{P}$ and $\mathbf{B}$ are specified, defines a linear system that needs to be solved for $\mathbf{x}$.

2.2 Invertibility of a Matrix

For this solution method, the use of an inverse matrix is required.

Theorem 1. *[3] Let $\mathcal{P}$ be a square matrix. If R is a square matrix satisfying $R\mathcal{P} = I$, then $R = \mathcal{P}^{-1}$.*

Proof. Assume that $R\mathcal{P} = I$. If we can show that $\mathcal{P}$ is invertible, the proof can be completed by multiplying $R\mathcal{P} = I$ on both sides by $\mathcal{P}^{-1}$ to obtain

$$\begin{aligned} R\mathcal{P}\mathcal{P}^{-1} &= I\mathcal{P}^{-1} \\ RI &= I\mathcal{P}^{-1} \\ R &= \mathcal{P}^{-1} \end{aligned}$$

□

The matrix is invertible if and only if the determinant has a multiplicative inverse, defined in Lemma 1.

Lemma 1. *[3] Let M be a ring with unity. Let $\mathcal{P}$ be a square matrix of order n. Then, $\mathcal{P} \in M^{n\times n}$ is invertible if and only if its determinant is invertible in M.*

Proof. Let $\mathcal{P}$ be invertible with $M = \mathcal{P}^{-1}$. Let 1_M denote the unity of M. Let I_n denote the unit matrix of order n. Then,

$$\begin{aligned} 1_M &= det(I_n) \\ &= det(\mathcal{P}M) \\ &= det(\mathcal{P})det(M) \end{aligned}$$

This shows that $det(M) = \frac{1}{det(\mathcal{P})}$. □

2.3 Solving the System of Equations by Matrix Inversion

In addition to the well-known methods of Gaussian and Gauss-Jordan elimination, the following theorem introduces an additional approach to solving specific linear problems.

Theorem 2. *[3] If $\mathcal{P}$ is an invertible $n \times n$ matrix, then for each $n \times 1$ matrix $\mathbf{B}$, the system of equations $\mathcal{P}\mathbf{x} = \mathbf{B}$ has exactly one solution, namely, $\mathbf{x} = \mathcal{P}^{-1}\mathbf{B}$.*

Proof. Since $\mathcal{P}(\mathcal{P}^{-1}\mathbf{B}) = \mathbf{B}$, it follows that $\mathbf{x} = \mathcal{P}^{-1}\mathbf{B}$ is a solution of $\mathcal{P}\mathbf{x} = \mathbf{B}$. To show that this is the only solution, we will assume that $\mathbf{x}_0$ is an arbitrary solution and then show that $\mathbf{x}_0$ must be the solution $\mathcal{P}^{-1}\mathbf{B}$.
If $\mathbf{x}_0$ is any solution, then $\mathcal{P}\mathbf{x}_0 = \mathbf{B}$. Multiplying both sides by $\mathcal{P}^{-1}$, we obtain $\mathbf{x}_0 = \mathcal{P}^{-1}\mathbf{B}$. □

3 Preliminaries on Multivariate Quadratic Polynomials

This section reviews the basic terminology and cryptographic primitives utilised in multivariate cryptography.

3.1 Matrix Representation

Definition 1. (Multivariate Quadratic Polynomials) [10]
Let $\mathbb{F}_q$ be a finite field with q elements. We denote m as the number of equations and n as the number of variables. A system $\mathcal{P} = (p_1, \ldots, p_m)$ of multivariate quadratic polynomials is defined as

$$
\begin{aligned}
p_1(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{(1)} \cdot x_i x_j + \sum_{i=1}^{n} t_i^{(1)} \cdot x_i + t_0^{(1)} \\
p_2(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{(2)} \cdot x_i x_j + \sum_{i=1}^{n} t_i^{(2)} \cdot x_i + t_0^{(2)} \\
&\vdots \\
p_m(x_1, \ldots, x_n) &= \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{(m)} \cdot x_i x_j + \sum_{i=1}^{n} t_i^{(m)} \cdot x_i + t_0^{(m)}.
\end{aligned}
$$

Definition 2. Let $\mathcal{P}(x_1, \ldots, x_n)$ be a system of multivariate quadratic polynomials.

a) The **lexicographical ordering** of monomials is defined as the order in which the monomials (without the coefficient) would exist in terms of words in an alphabet $x_1, x_2, \ldots, x_n$ letters [19].

b) We define the **chosen-lexicographical ordering** of monomials, which are listed from two variables to a single variable and are subject to the priority of solving a variable.

Throughout this research, Definition 2 b) will be utilized to convert the system of polynomials into the Macaulay matrix, which will then be solved by Gaussian elimination.

The chosen-lexicographical ordering of monomials emphasizes the priority of solving a variable. As an illustration, assume x_3 to be eliminated first. It is possible to obtain a univariate polynomial in terms of x_3 with a degree of at most d. The univariate polynomial over the finite field is then solved. As a result, the possible value(s) of x_3 are obtained. To that end, we will now substitute x_3 to obtain polynomials with fewer variables.

Definition 3. [10] (Multivariate Quadratic Problem) Let $\mathbb{F}_q$ be a finite field with q elements. Given a system $\mathcal{P} = (p_1(x), \ldots, p_m(x))$ of m multivariate quadratic polynomials in n variables, find a vector $\mathbf{x} = (x_1, \ldots, x_n)$ such that

$$p_1(\mathbf{x}) = \ldots = p_m(\mathbf{x}) = 0.$$

Theorem 3. *[10] Let $\mathbb{F}$ be a finite field with q elements and degree $d < q$. Then, there exist $\binom{n+d-1}{d}$ monomials of degree d in $\mathbb{F}[x_1, \ldots, x_n]$. The number of monomials of degree $\leq d$ in $\mathbb{F}[x_1, \ldots, x_n]$ is given by $\binom{n+d}{d}$.*

Proof. 1. The number of monomials of degree d is obtain by choosing d out of n element of $x_1, \ldots, x_n$, with repetition.

2. The elements in polynomial of degree $\leq d$ are elements from the set $\{x_1, \ldots, x_n, 1\}$, with repetition.

□

Theorem 4. *The number of monomials of degree $\leq d$ in $\mathbb{F}_q$ with q elements is given by $\binom{n+d}{d}$. The number of monomials with degree d with different elements is given by $\binom{n}{d}$.*

Proof. 1. The total number of monomials of degree $\leq d$ is according to n elements from the set $\{x_1, \ldots, x_n, 1\}$, with repetition.

2. The total number of monomials of degree $\leq d$ is according to n elements from the set $\{x_1, \ldots, x_n, 1\}$, without repetition.

□

A system of MQ polynomial equations is resolved by transforming the polynomials in the Macaulay matrix known as matrix M, which is constructed from equations $(p_1(x), \ldots, p_m(x)) = 0$ (Definition 3) and reduced using the Gaussian elimination procedure. The number of column vectors in the Macaulay matrix M is the number of the monomials of degree $\leq d$, which is $\binom{n+d}{d}$.

3.2 General Workflow of MPKC

Multivariate public key cryptosystem based on the MQP is constructed from an invertible quadratic map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ and two invertible affine (or linear) maps $\mathcal{S} : \mathbb{F}^m \to \mathbb{F}^m$ and $\mathcal{T} : \mathbb{F}^n \to \mathbb{F}^n$. The *public key* is in the form of $\mathcal{P} = \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}$ where $\mathcal{S}, \mathcal{F}$ and $\mathcal{T}$ are the *private keys.*

3.2.1 General Encryption Scheme ($m \geq n$)

To create a comprehensive encryption scheme, it is necessary to determine the corresponding secret affine maps, $\mathcal{S}$ and $\mathcal{T}$, and the central map, $\mathcal{F}$. The fundamental encryption scheme, as outlined in [10], can be succinctly expressed as follows.

Key Generation:

$$\mathcal{S} = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1m} \\ s_{21} & s_{22} & \cdots & s_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m1} & s_{m2} & \cdots & s_{mm} \end{pmatrix}$$

$$\mathcal{T} = \begin{pmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n1} & t_{n2} & \cdots & t_{nn} \end{pmatrix}$$

$$\mathcal{F} = \begin{cases} f^{(1)}(x_1, x_2, \ldots, x_n) & = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^{(1)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(1)} x_i + \sum_{i=1}^{n} \gamma_i^{(1)} \\ f^{(2)}(x_1, x_2, \ldots, x_n) & = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^{(2)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(2)} x_i + \sum_{i=1}^{n} \gamma_i^{(2)} \\ & \vdots \\ f^{(m)}(x_1, x_2, \ldots, x_n) & = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^{(m)} x_i x_j + \sum_{i=1}^{n} \beta_i^{(m)} x_i + \sum_{i=1}^{n} \gamma_i^{(m)} \end{cases}$$

Encryption: To encrypt a message $\mathbf{x} \in \mathbb{F}^n$, one simply computes $\mathcal{P}(\mathbf{x}) = \mathbf{z}$.

$$\begin{aligned} \mathcal{P} &= \mathcal{S} \circ \mathcal{F} \circ \mathcal{T}(\mathbf{x}) \\ \mathbf{w} &= \mathcal{T}(\mathbf{x}) \\ \mathbf{y} &= \mathcal{F}(\mathbf{w}) \\ \mathbf{z} &= \mathcal{S}(\mathbf{y}) \end{aligned}$$

The *ciphertext* of the message $\mathbf{x}$ is $\mathbf{z} \in \mathbb{F}^m$.

Decryption: To decrypt the ciphertext $\mathbf{z} \in \mathbb{F}^m$, one computes $\mathcal{P}^{-1} = \mathbf{x}$ recursively.

$$\begin{aligned} \mathbf{y} &= \mathcal{S}^{-1}(\mathbf{z}) \\ \mathbf{w} &= \mathcal{F}^{-1}(\mathbf{y}) \\ \mathbf{x} &= \mathcal{T}^{-1}(\mathbf{w}) \end{aligned}$$

$\mathbf{x} \in \mathbb{F}^n$ is the *plaintext* corresponding to the ciphertext $\mathbf{z}$.

Proof of correctness:

$$\begin{aligned} \mathcal{T}^{-1} \circ \mathcal{F}^{-1} \circ \mathcal{S}^{-1}(\mathbf{z}) &= \mathcal{T}^{-1}(\mathcal{F}^{-1}(\mathcal{S}^{-1}(\mathbf{z}))) \\ &= \mathcal{T}^{-1}(\mathcal{F}^{-1}(\mathbf{y})) \\ &= \mathcal{T}^{-1}(\mathbf{w}) \\ &= \mathbf{x} \end{aligned}$$

3.2.2 Standard Attacks

There are two types of standard attacks against multivariate public key schemes.

1. **Direct Attack:** This kind of attack concentrates on solving the public equation $\mathcal{P}(\mathbf{x}) = \mathbf{z}$ as an instance of the MQP directly. The examples of direct attack are F_4 and F_5 algorithm (Gröbner basis) [16, 20] and XL algorithm [7, 27].

2. **Structural Attack:** A structural attack requires the unique structure of the central map of a multivariate cryptography scheme to endeavour to recover the private key—for example, a Linearization equation attack, a MinRank attack, or a Differential attack.

4 New Method for Solving MQP

In this section, we will present the determinant function, which assigns a real integer to a square matrix. It is a real-valued function of a matrix variable. The determinant will have significant implications for the theory of systems of linear equations. It will also provide us with a precise formula for the inverse of a matrix that can be inverted.

Definition 4. [3] If $\mathcal{P}$ is a square matrix, then the minor of entry a_{ij} is denoted by M_{ij} and is defined to be the determinant of the submatrix that remains after the ith row and jth column are deleted from $\mathcal{P}$. The number $(-1)^{i+j}M_{ij}$ is denoted by C_{ij} and is called the cofactor of entry a_{ij}.

In a more general sense, we define the determinant of a $n \times n$ matrix as

$$det(\mathcal{P}) = a_{11}C_{11} + a_{12}C_{12} + \cdots + a_{1n}C_{1n}.$$

This process of determining the value of $\det(\mathcal{P})$ (in Definition 4) is referred to as cofactor expansion along the first row of matrix $\mathcal{P}$.

Definition 5. The determinant of an $n \times n$ matrix $\mathcal{P}$ can be computed by multiplying the entries in any row (or column) by their cofactors and adding the resulting products; that is, for each $1 \leq i \leq n$ and $1 \leq j \leq n$,

$$det(\mathcal{P}) = a_{1j}C_{1j} + a_{2j}C_{2j} + \cdots + a_{nj}C_{nj}$$
(cofactor expansion along the jth column)

and

$$det(\mathcal{P}) = a_{i1}C_{i1} + a_{i2}C_{i2} + \cdots + a_{in}C_{in}$$
(cofactor expansion along the ith row)

Note that we may choose any row or any column.

Definition 6. [3] If $\mathcal{P}$ is any $n \times n$ matrix and C_{ij} is the cofactor of a_{ij}, then the matrix

$$\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

is called the matrix of cofactor from $\mathcal{P}$. The transpose of this matrix is called the adjoint of $\mathcal{P}$ and is denoted by $\text{adj}(\mathcal{P})$.

Definition 7. [17] Let $\mathcal{P}$ be an $n \times n$ matrix with integer entries, and let q be an integer modulo. If $gcd(det(\mathcal{P}, q)) = 1$, then $\mathcal{P}$ is invertible modulo q. This means there exists an $n \times n$ matrix M with integer entries such that $\mathcal{P}M = I_n \pmod q$, where I_n is the $n \times n$ identity matrix.

Remark 1. Formally, if $gcd(det(\mathcal{P}), q) = 1$, then there exists an $n \times n$ matrix M such that $\mathcal{P}M = I_n \pmod q$ or $\mathcal{P}$ is invertible modulo q.

To obtain the formula for the inverse of an invertible matrix, it is essential to use the important fact that a square matrix $\mathcal{P}$ is invertible if and only its determinant, $det(\mathcal{P})$ is non-zero.

Theorem 5. *(Inverse of a Matrix using its Adjoint)[3]. If $\mathcal{P}$ is an invertible matrix, then*

$$\mathcal{P}^{-1} = \frac{1}{det(\mathcal{P})} adj(\mathcal{P})$$

Proof. We first show that

$$\mathcal{P}adj(\mathcal{P}) = det(\mathcal{P})I$$

Consider the product

$$\mathcal{P}adj(\mathcal{P}) = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} C_{11} & C_{21} & \cdots & C_{j1} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{2n} & \cdots & C_{n1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_{1n} & C_{2n} & \cdots & C_{jn} & \cdots & C_{nn} \end{bmatrix}$$

The entry in the ith row and jth column of the product $\mathcal{P}adj(\mathcal{P})$ is

$$a_{i1}C_{j1} + a_{i2}C_{j2} + \cdots + a_{in}C_{jn} \tag{1}$$

If $i = j$, then (1) is the cofactor expansion of $det(\mathcal{P})$ along the ith row of $\mathcal{P}$, and if $i \neq j$, then the $a's$ and the cofactors come from different rows of $\mathcal{P}$, so the value of (1) is zero. Therefore,

$$\mathcal{P}adj(\mathcal{P}) = \begin{bmatrix} det(\mathcal{P}) & 0 & \cdots & 0 \\ 0 & det(\mathcal{P}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & det(\mathcal{P}) \end{bmatrix} = det(\mathcal{P})I \tag{2}$$

Since $det(\mathcal{P})$ is invertible, thus $det(\mathcal{P}) \neq 0$. Therefore, Equation (2) can be written as

$$\frac{1}{det(\mathcal{P})}[\mathcal{P}adj(\mathcal{P})] = I, \text{ or } \mathcal{P}[\frac{1}{det(\mathcal{P})} adj(\mathcal{P})] = I$$

Multiplying both sides by $\mathcal{P}^{-1}$ yields

$$\mathcal{P}^{-1} = \frac{1}{det(\mathcal{P})} adj(\mathcal{P})$$

□

Breaking the MQ polynomial is to obtain the value of candidates for plaintext from a given public system $\mathcal{P}$ and ciphertext z.

Lemma 2. *Let $\mathcal{P}$ be $r \times s$ matrix. If $r = s$, then $\mathcal{P}$ is invertible.*

Proof. **Case 1:** $r \neq s$ and $r < s$.
Given

$$[\mathcal{P}]^{r\times s} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rs} \end{bmatrix}$$

Then, we add k additional rows so that $(r + k) = s$, that is, a square matrix $\overline{\mathcal{P}}$ is formed.

$$\overline{\mathcal{P}} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rs} \\ c_{(r+1)1} & c_{(r+1)2} & \cdots & c_{(r+1)s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(r+k)1} & c_{(r+k)2} & \cdots & c_{(r+k)s} \end{bmatrix}$$

Case 2: $r \neq s$ and $r > s$.
Given

$$[\mathcal{P}]^{r\times s} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{r1} & c_{r2} & \cdots & c_{rs} \end{bmatrix}$$

We exclude excess k rows such that $(r - k) = s$, that is, a square matrix $\overline{\mathcal{P}}$ is formed.

$$\overline{\mathcal{P}} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(r-k)1} & c_{(r-k)2} & \cdots & c_{(r-k)s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(r-1)1} & c_{(r-1)2} & \cdots & c_{(r-1)s} \\ c_{r1} & c_{r2} & \cdots & c_{rs} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1s} \\ c_{21} & c_{22} & \cdots & c_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(r-k)1} & c_{(r-k)2} & \cdots & c_{(r-k)s} \end{bmatrix}$$

Now, this square matrix $\overline{\mathcal{P}}$ is a ring with unity. Then, by Lemma 1 and Theorem 1, this $\overline{\mathcal{P}}$ is invertible for both Case 1 and Case 2.

□

4.1 Technique of solving the MQP

The procedure for solving MQ polynomials based on our method is shown below in Algorithm 1.

Algorithm 1 Solving the Multivariate Quadratic polynomials

Input: The quadratic polynomial $\mathcal{P} = (p_1(\mathbf{x}), \ldots, p_m(\mathbf{x}))$ in a finite field $\mathbb{F}_q$ with n variables of $(x_1, \ldots, x_n)$.
Output: The solution in $\mathbb{F}_q$ of the system of the equations $p_1(\mathbf{x}) = \ldots = p_m(\mathbf{x}) = 0$.

1: Transform the polynomial $\mathcal{P}$ into the matrix equation $\mathcal{P}\mathbf{x} = \mathbf{B}$ (see Section 2.1).
2: **if** the matrix $\mathcal{P}$ is in square matrix **then**
3: the system can be solved using the theorem in Section 2.3.
4: **else if** the matrix $\mathcal{P}$ is a non-square matrix, and the number of rows is less than columns **then**
5: additional rows must be added to match the number of columns where each element in the rows must not be identical or proportional as any other rows.
6: **else if** the matrix $\mathcal{P}$ is a non-square matrix, and the number of rows is greater than columns **then**
7: omit the excess rows to form a square matrix, and solve it as in Section 2.3.
8: **end if**

The efficient procedure for identifying additional rows ensures that no rows or columns are identical or proportional.

Theorem 6. *[3] If $\mathcal{P}$ is a square matrix with two proportional rows and two proportional columns, then $det(\mathcal{P}) = 0$.*

Proof. See [3]. □

Theorem 7. *Let $\mathcal{P} = (p_1(\mathbf{x}), \ldots, p_m(\mathbf{x}))$ be a quadratic polynomial in a finite field $\mathbb{F}_q$ with n variables of $(x_1, \ldots, x_n)$. Then, a system of the equations $p_1(\mathbf{x}) = \ldots = p_m(\mathbf{x}) = 0$ admits a solution in $\mathbb{F}_q$.*

Proof. We begin by considering the system of equation $\mathcal{P}$ in two cases.
Case 1: $\mathcal{P}$ is a square matrix.
If $\mathcal{P}$ is a square matrix, then the system of equation $\mathcal{P}$ is solved as in Section 2.3.

Case 2: $\mathcal{P}$ is non-square matrix.
If $\mathcal{P}$ is not a square matrix, either the number of rows less than columns or the number of rows greater columns, then the system of equations $\mathcal{P}$ can be solved by Lemma 2 and followed by Section 2.3.

□

5 Toy Example

For illustrative purposes, an example where public system $\mathcal{P}$ consisting of a multivariate quadratic polynomial over the field $\mathbb{F}_q$ is provided, featuring randomly chosen coefficients. We give an example for the case where the matrix $\mathcal{P}$ is non-square. It can be categorized into two types: the number of rows exceeds the number of columns, or the number of rows is less than the number of columns.

1. If the number of rows is less than the number of columns, additional rows must be added to match the number. Furthermore, each element in the rows must be coprime or undergo an arithmetic operation that ensures they are neither identical nor proportional to any other row.

2. If the number of rows exceeds the number of columns, omit the excess rows to form a square matrix and then solve it using Theorem 2.

Example 1. Suppose the message vector $\mathbf{x} = (x_1, x_2, x_3) = (2, 3, 5)$ is encrypted into the ciphertext $\mathbf{z} = (6, 2, 1, 6, 4, 1)$. Then, we are given the public system $\mathcal{P}(\mathbf{x})$ over the field $\mathbb{F}_7$:

$$\mathcal{P}(\mathbf{x}) = \begin{cases} 2x_1x_2 + 1x_1x_3 + 5x_2x_3 + 4x_1^2 + 4x_2^2 + 2x_2 + 2x_3^2 + 5x_3 \\ 2x_1x_2 + 1x_1x_3 + 4x_2x_3 + 4x_1^2 + 3x_2^2 + 2x_2 + 2x_3^2 + 2x_3 \\ 2x_1x_2 + 4x_1x_3 + 6x_1^2 + 1x_1 + 6x_2^2 + 4x_2 + 4x_3^2 + 6x_3 \\ 1x_1x_2 + 3x_2x_3 + 1x_1^2 + 6x_2^2 + 6x_3^2 + 4x_3 \\ 2x_1x_2 + 2x_1x_3 + 5x_2x_3 + 5x_1 + 3x_2^2 + 3x_2 + 3x_3^2 \\ 3x_1x_2 + 2x_1x_3 + 1x_2x_3 + 1x_1^2 + 1x_1 + 4x_2 + 2x_3^2 + 4x_3 \end{cases}$$

Then, we conduct the following procedure.
Transforming the system $\mathcal{P}(\mathbf{x})$ and ciphertext vector into the single matrix equation

$$\begin{bmatrix} 2x_1x_2 + 1x_1x_3 + 5x_2x_3 + 4x_1^2 + 4x_2^2 + 2x_2 + 2x_3^2 + 5x_3 \\ 2x_1x_2 + 1x_1x_3 + 4x_2x_3 + 4x_1^2 + 3x_2^2 + 2x_2 + 2x_3^2 + 2x_3 \\ 2x_1x_2 + 4x_1x_3 + 6x_1^2 + 1x_1 + 6x_2^2 + 4x_2 + 4x_3^2 + 6x_3 \\ 1x_1x_2 + 3x_2x_3 + 1x_1^2 + 6x_2^2 + 6x_3^2 + 4x_3 \\ 2x_1x_2 + 2x_1x_3 + 5x_2x_3 + 5x_1 + 3x_2^2 + 3x_2 + 3x_3^2 \\ 3x_1x_2 + 2x_1x_3 + 1x_2x_3 + 1x_1^2 + 1x_1 + 4x_2 + 2x_3^2 + 4x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 1 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

The $m \times 1$ matrix on the left side of this equation can be written as $\mathcal{P}\mathbf{x} = \mathbf{z}$, where

$$\begin{bmatrix} 2 & 1 & 5 & 4 & 0 & 4 & 2 & 2 & 5 \\ 2 & 1 & 4 & 4 & 0 & 3 & 2 & 2 & 2 \\ 2 & 4 & 0 & 6 & 1 & 6 & 4 & 4 & 6 \\ 1 & 0 & 3 & 1 & 0 & 6 & 0 & 6 & 4 \\ 2 & 2 & 5 & 0 & 5 & 3 & 3 & 3 & 0 \\ 3 & 2 & 1 & 1 & 1 & 0 & 4 & 2 & 4 \end{bmatrix} \begin{bmatrix} x_1x_2 \\ x_1x_3 \\ x_2x_3 \\ x_1^2 \\ x_1 \\ x_2^2 \\ x_2 \\ x_3^2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 1 \\ 6 \\ 4 \\ 1 \end{bmatrix}$$

At this point, we observed that the matrix $\mathcal{P}$ is non-square. Therefore, we added three rows to transform matrix $\mathcal{P}$ into a square matrix. The elements in the additional rows are random numbers and not identical or proportional to any other rows. The new matrix $\mathcal{P}$ is as follows.

$$\begin{bmatrix} 2 & 1 & 5 & 4 & 0 & 4 & 2 & 2 & 5 \\ 2 & 1 & 4 & 4 & 0 & 3 & 2 & 2 & 2 \\ 2 & 4 & 0 & 6 & 1 & 6 & 4 & 4 & 6 \\ 1 & 0 & 3 & 1 & 0 & 6 & 0 & 6 & 4 \\ 2 & 2 & 5 & 0 & 5 & 3 & 3 & 3 & 0 \\ 3 & 2 & 1 & 1 & 1 & 0 & 4 & 2 & 4 \\ 4 & 3 & 2 & 0 & 0 & 2 & 0 & 1 & 2 \\ 4 & 6 & 1 & 2 & 1 & 1 & 2 & 2 & 1 \\ 2 & 5 & 1 & 6 & 3 & 4 & 0 & 3 & 3 \end{bmatrix} \begin{bmatrix} x_1x_2 \\ x_1x_3 \\ x_2x_3 \\ x_1^2 \\ x_1 \\ x_2^2 \\ x_2 \\ x_3^2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 1 \\ 6 \\ 4 \\ 1 \\ 4 \\ 4 \\ 2 \end{bmatrix}$$

Now, we observed that the new $\mathcal{P}$ is a square matrix [1]. The determinant of the new matrix

[1] If the system of equations $\mathcal{P}$ originally gives a square matrix, then the matrix $\mathcal{P}$ can be solves directly using Theorem 2.

$\mathcal{P}$ is non-zero and coprime to the modulus $q = 7$, therefore $\mathcal{P}$ is invertible.

$$\mathcal{P}^{-1} = \begin{bmatrix} 5 & 4 & 1 & 3 & 1 & 5 & 5 & 2 & 5 \\ 3 & 5 & 6 & 4 & 6 & 0 & 4 & 6 & 0 \\ 2 & 6 & 3 & 3 & 5 & 4 & 3 & 2 & 5 \\ 5 & 2 & 2 & 0 & 0 & 0 & 5 & 3 & 3 \\ 6 & 4 & 4 & 3 & 5 & 4 & 6 & 5 & 4 \\ 5 & 4 & 6 & 0 & 3 & 2 & 1 & 2 & 1 \\ 6 & 3 & 0 & 4 & 4 & 0 & 5 & 3 & 4 \\ 3 & 3 & 1 & 0 & 0 & 0 & 1 & 6 & 0 \\ 5 & 1 & 4 & 6 & 2 & 5 & 1 & 1 & 5 \end{bmatrix}$$

Thus, the solution of this system is

$$\mathbf{x} = \mathcal{P}^{-1}\mathbf{z} = \begin{bmatrix} 5 & 4 & 1 & 3 & 1 & 5 & 5 & 2 & 5 \\ 3 & 5 & 6 & 4 & 6 & 0 & 4 & 6 & 0 \\ 2 & 6 & 3 & 3 & 5 & 4 & 3 & 2 & 5 \\ 5 & 2 & 2 & 0 & 0 & 0 & 5 & 3 & 3 \\ 6 & 4 & 4 & 3 & 5 & 4 & 6 & 5 & 4 \\ 5 & 4 & 6 & 0 & 3 & 2 & 1 & 2 & 1 \\ 6 & 3 & 0 & 4 & 4 & 0 & 5 & 3 & 4 \\ 3 & 3 & 1 & 0 & 0 & 0 & 1 & 6 & 0 \\ 5 & 1 & 4 & 6 & 2 & 5 & 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \\ 6 \\ 4 \\ 1 \\ 4 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 1 \\ 4 \\ 2 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Therefore, we obtain the solution $x_1 = 2$, $x_2 = 3$ and $x_3 = 5$.

In cases where the number of rows exceeds the number of columns, the excess row(s) can be omitted to form a square matrix.

Example 2. Suppose the message vector $\mathbf{x} = (x_1, x_2) = (4, 7)$ is encrypted into the ciphertext $\mathbf{x} = (2, 11, 12, 5, 7, 1, 0)$. Then, we are given the public system $\mathcal{P}(\mathbf{x})$ over the field $\mathbb{F}_{17}$:

$$\mathcal{P}(\mathbf{x}) = \begin{cases} 1x_1x_2 + 6x_1^2 + 4x_1 + 2x_2^2 + 10x_2, \\ 13x_1x_2 + 12x_1^2 + 2x_1 + 11x_2^2 + 14x_2, \\ 1x_1x_2 + 1x_1^2 + 6x_1 + 15x_2^2 + 6x_2, \\ 2x_1x_2 + 12x_1^2 + 11x_1 + 8x_2^2 + 5x_2, \\ 14x_1x_2 + 0x_1^2 + 1x_1 + 14x_2^2 + 14x_2, \\ 6x_1x_2 + 8x_1^2 + 12x_1 + 12x_2^2 + 14x_2, \\ 3x_1x_2 + 8x_1^2 + 12x_1 + 12x_2^2 + 10x_2 \end{cases}$$

Subsequently, we perform the following procedure.
Transforming the system $\mathcal{P}(\mathbf{x})$ and the ciphertext vector into a single matrix equation

$$\begin{bmatrix} 1x_1x_2 + 6x_1^2 + 4x_1 + 2x_2^2 + 10x_2 \\ 13x_1x_2 + 12x_1^2 + 2x_1 + 11x_2^2 + 14x_2 \\ 1x_1x_2 + 1x_1^2 + 6x_1 + 15x_2^2 + 6x_2 \\ 2x_1x_2 + 12x_1^2 + 11x_1 + 8x_2^2 + 5x_2 \\ 14x_1x_2 + 0x_1^2 + 1x_1 + 14x_2^2 + 14x_2 \\ 6x_1x_2 + 8x_1^2 + 12x_1 + 12x_2^2 + 14x_2 \\ 3x_1x_2 + 8x_1^2 + 12x_1 + 12x_2^2 + 10x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 11 \\ 12 \\ 5 \\ 7 \\ 1 \\ 0 \end{bmatrix}$$

Transform the matrix on the left-hand side into the form $\mathcal{P}\mathbf{x} = \mathbf{z}$.

$$\begin{bmatrix} 1 & 6 & 4 & 2 & 10 \\ 13 & 12 & 2 & 11 & 14 \\ 1 & 1 & 6 & 15 & 6 \\ 2 & 12 & 11 & 8 & 5 \\ 14 & 0 & 1 & 14 & 14 \\ 6 & 8 & 12 & 12 & 14 \\ 3 & 8 & 12 & 12 & 10 \end{bmatrix} \begin{bmatrix} x_1x_2 \\ x_1^2 \\ x_1 \\ x_2^2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 11 \\ 12 \\ 5 \\ 7 \\ 1 \\ 0 \end{bmatrix}$$

At this stage, we noticed that the matrix $\mathcal{P}$ is non-square matrix with two extra rows. Therefore, we exclude these extra rows to transform $\mathcal{P}$ into a square matrix. Then, the new matrix $\mathcal{P}$ is as follows.

$$\begin{bmatrix} 1 & 6 & 4 & 2 & 10 \\ 13 & 12 & 2 & 11 & 14 \\ 1 & 1 & 6 & 15 & 6 \\ 2 & 12 & 11 & 8 & 5 \\ 14 & 0 & 1 & 14 & 14 \end{bmatrix} \begin{bmatrix} x_1x_2 \\ x_1^2 \\ x_1 \\ x_2^2 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 11 \\ 12 \\ 5 \\ 7 \end{bmatrix}$$

The determinant of the new matrix $\mathcal{P}$ is non-zero and coprime to the modulus 17; therefore, $\mathcal{P}$ is invertible.

$$\mathcal{P}^{-1} = \begin{bmatrix} 7 & 6 & 2 & 13 & 9 \\ 15 & 10 & 10 & 3 & 14 \\ 1 & 5 & 5 & 4 & 15 \\ 7 & 11 & 7 & 9 & 13 \\ 9 & 13 & 4 & 2 & 11 \end{bmatrix}$$

Thus, the solution of this system is

$$\mathbf{x} = \mathcal{P}^{-1}\mathbf{z} = \begin{bmatrix} 7 & 6 & 2 & 13 & 9 \\ 15 & 10 & 10 & 3 & 14 \\ 1 & 5 & 5 & 4 & 15 \\ 7 & 11 & 7 & 9 & 13 \\ 9 & 13 & 4 & 2 & 11 \end{bmatrix} \begin{bmatrix} 2 \\ 11 \\ 12 \\ 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 11 \\ 16 \\ 4 \\ 15 \\ 7 \end{bmatrix}$$

Therefore, we obtain the solution $x_1 = 4$ and $x_2 = 7$.

Conclusion

We presented a new approach for solving the Multivariate Quadratic (MQ) problem using systems of non-linear equations and matrices. Our finding enables one to identify the matrix invertibility of the public key by determining its non-zero determinant and coprimality to the modulus. This strategy transformed the system of MQ polynomials into a system represented as $\mathcal{P}\mathbf{x} = \mathbf{B}$. We considered two distinct scenarios: one where the matrix $\mathcal{P}$ is square and another where it is non-square. For the square matrix case, the solution is straightforwardly obtained by evaluating the inverse of $\mathcal{P}$ and multiplying it by the vector $\mathbf{B}$. For the non-square matrix, the approach involves adding more rows to the matrix to achieve a square form when there are fewer rows than columns or reducing the matrix by omitting excess rows when there are more rows than columns. This transformation not only simplifies the problem but also utilizes the wide range of tools and methods available for solving non-linear systems. Therefore, it enhances the efficiency and applicability of the solution in various domains, including cryptography and computational mathematics. This exploration opens new opportunities for further research and practical applications of solving MQ problems through linear algebraic techniques.

Acknowledgements

The first author wishes to acknowledge the invaluable support from the Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM), the Ministry of Higher Education (MOHE), and Universiti Teknologi MARA (UiTM) for providing the opportunity to conduct this research.

References

[1] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the first round of the nist post-quantum cryptography standardization process. 2019.

[2] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[3] Howard Anton and Chris Rorres. *Elementary linear algebra: applications version*. John Wiley & Sons, 2013.

[4] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*, volume 1. Springer, 2008.

[5] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[6] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 392–407. Springer, 2000.

[7] Nicolas T Courtois and Jacques Patarin. About the xl algorithm over gf (2). In *Topics in Cryptology—CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003 Proceedings*, pages 141–157. Springer, 2003.

[8] Nicolas T Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings 8*, pages 267–287. Springer, 2002.

[9] Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.

[10] Jintai Ding, Albrecht Petzoldt, and Dieter S. Schmidt. Multivariate public key cryptosystems. 2020.

[11] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.

[12] Jean Charles Faugere. A new efficient algorithm for computing gröbner bases without reduction to zero (f 5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[13] Jean-Charles Faugere and Gwénolé Ars. *Comparison of XL and Gröbner basis algorithms over Finite Fields*. PhD thesis, INRIA, 2004.

[14] Aviezri S Fraenkel and Yaacov Yesha. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics*, 1(1-2):15–30, 1979.

[15] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. Freeman, 1979.

[16] Takuma Ito, Yuta Hoshi, Naoyuki Shinohara, and Shigenori Uchiyama. Polynomial selection of f4 for solving the mq problem. *JSIAM Letters*, 14:135–138, 2022.

[17] Fausto Abraham Jacques-García, Daniel Uribe-Mejía, Gonzalo Macías-Bobadilla, and Ricardo Chaparro-Sánchez. On modular inverse matrices a computation approach. *South Florida Journal of Development*, 3(3):3100–3111, 2022.

[18] Natalia Kharchenko. *Lattice algorithms and lattice-based cryptography*. PhD thesis, Sorbonne Université, 2020.

[19] Neal Koblitz, Alfred J Menezes, Yi-Hong Wu, and Robert J Zuccherato. *Algebraic aspects of cryptography*, volume 198. Springer, 1998.

[20] Takashi Kurokawa, Takuma Ito, Naoyuki Shinohara, Akihiro Yamamura, and Shigenori Uchiyama. Selection strategy of f4-style algorithm to solve mq problems related to mpkc. *Cryptography*, 7(1):10, 2023.

[21] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[22] Ayman Wagih Mohsen, Ayman M Bahaa-Eldin, and Mohamed Ali Sobh. Lattice-based cryptography. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, pages 462–467. IEEE, 2017.

[23] Dustin Moody, Gorjan Alagic, Daniel C Apon, David A Cooper, Quynh H Dang, John M Kelsey, Yi-Kai Liu, Carl A Miller, Rene C Peralta, Ray A Perlner, et al. Status report on the second round of the nist post-quantum cryptography standardization process. 2020.

[24] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[25] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[26] Antonia Wachter-Zeh, Hannes Bartz, and Gianluigi Liva. *Code-Based Cryptography*. Springer, 2022.

[27] Lih-Chung Wang, Tzer-jen Wei, Jian-Ming Shih, Yuh-Hua Hu, and Chih-Cheng Hsieh. An algorithm for solving over-determined multivariate quadratic systems over finite fields. *Advances in Mathematics of Communications*, 18(1):55–90, 2024.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 48–63.

Current Status on Shor's Algorithm via Quantum Computing

Nur Azman Abu[1] and Zahari Mahad[2]

[1] Information Security Forensics and Computer Networking (INSFORNET), Centre for Advanced Computing Technology, Universiti Teknikal Malaysia, Melaka, Malaysia nura@utem.edu.my
[2] Laboratory of Cryptography, Analysis and Structure, Institute for Mathematical Research, Universiti Putra Malaysia, Selangor, Malaysia zaharimahad@upm.edu.my

Abstract. Since its inception in 1994, Shor's Algorithm has given an avenue to render RSA modulo factoring theoretically in real time. After 3 decades, this paper will investigation its practicality to factor a typical RSA modulo in polynomial time. A new breakthrough via a simple algorithm is still very much needed to harness quantum computing prowess in RSA factoring.

Keywords: Shor's Algorithm · Quantum Computer · Integer Factorization Problem

1 Introduction

Although any integer number has a unique decomposition into a product of primes, finding the prime factors is believed to be a hard problem. In fact, the security of our online transactions rests on the assumption that factoring integers with a thousand or more digits is practically impossible. This assumption was challenged in 1995 when Peter Shor proposed a polynomial-time quantum algorithm for RSA modulo factoring problem. Shor's Algorithm is arguably the most dramatic paradigm shift on how a quantum computing can change security perception on integer factoring to be tractable. A corner stone of the Shor's Algorithm is by far concentrated on its modular exponentiation that is the most computational component in time and space [10].

Current record on breaking RSA is around 800-bit modulo N as shown by a classical computer in RSA number challenge. An industrial standard operates on 2048-bit RSA modulo. A viable quantum computer using a practical Shor's Algorithm should be able to progressively factor a larger candidate of traditional RSA modulo N as the number of qubits grow inline with quantum computer progress. In the last 20 years, several reseachers have managed to factor an increasingly larger numbers. Unfortunately, these numbers carry only one or two bit security level which can be easily factor by a classic algorithm.

At the same time there is a need to build a qubit arithmetic calculator such as in [2] to handle an m-qubit quantum computer where $m < 80$ which matches current 64-bit CPU. This calculator can be running in sequential, parallel or distributed. At the same time, Intel has built a quantum processor called Tunnel Falls that will offer 12-qubit computing prowess. Intel believes its control chips and chip interconnect technology will be necessary parts of an overall system.

Scaling up quantum computers to attain substantial speedups over classical computing requires a significant fault tolerance [26]. Conventionally, protocols for fault-tolerant quantum computation demand excessive space overheads by using many physical qubits for each logical qubit.

One bit of error in running a secure cryptographic operation will produce an unrecoverable wrong output. Using plausible physical assumptions for large-scale planar grid of

qubits with nearest-neighbour connectivity, a characteristic physical gate error rate of 10^{-3}, a surface code cycle time of 1 microsecond, and a reaction time of 10 microseconds, [11] has estimated a need to use $3m + 0.002m \log_2 m$ logical qubits, $0.3m^3 + 0.0005m^3 \log_2 m$ Toffoli gates, and $500m^2 + m^2 \log_2 m$ measurement depth to factor an m-bit RSA modulo.

Accordingly, under current quantum computing, it is achievable to factor 16-bit RSA modulo using 48 logical qubits, 1237 Toffoli gates and 129,024 measurement depth. Let alone to factor 2048-bit RSA modulo, they project basic requirement of a mere 6,189 logical qubits, 2,624,225,018 Toffoli gates and 2,143,289,344 measurement depth. Six thousand plus qubits sounds reachable in the foreseeable future. A two billion gate setup will consume all a significant RnD fund available to a quantum computing research group.

According to [3], $2m + 3$ qubits will be sufficient to factor m-bit RSA modulo. A quantum arithmetic is constructed from Toffoli gates and quantum Fourier transform (QFT) [23]. It consists of carry-based modular addition, multiplication, and exponentiation circuits. Then [3] has introduced a circuit for Shor's Algorithm using QFT via $2m + 3$ qubits and $O(m^3 \log m)$ elementary quantum gates in a depth of $O(m^3)$ to implement the factorization algorithm. The circuit is computable in polynomial time on a classical computer and is completely general as it does not rely on any property of the number to be factored.

Quantum comparators and modular arithmetic are fundamental in a quantum algorithm. [28] propose a quantum–classical comparator based on the QFT using operators upon which only require up to one ancilla qubit. This achievement will cut down to $2m + 1$ qubits sufficient to factor m-bit RSA modulo.

[24] has presented a quantum algorithm to factor products of prime numbers exploiting a Grover search that uses only $2m-5$ qubits and elementary quantum arithmetic operations. Through large scale numerical simulations, Grover based factoring requires quadratically fewer iteration steps than previous digital adiabatic algorithms.

[4] focuses on the optimization of the number of logical qubits in Shor's quantum factoring algorithm using a Residue Number System(RNS) requiring only $m + o(m)$ qubits with gate count $O(m^3)$ and $O(m^2 \log^3 m)$ depth in the case of an m-bit RSA modulus. Preliminary logical resource estimates suggest that this circuit could be engineered to use less than 1700 qubits and Toffoli gates, and require 60 independent runs to factor an RSA-2048 instance.

Latest estimate by Regev in 2024 [19]) on An Efficient Quantum Factoring Algorithm, state that an m-bit integer can be factored by independently running a quantum circuit with $O(m^{3/2})$ gates in $m^{1/2} + 4$ running times. This analysis has lowered down the celebrated Shor's Algorithm which allows to factorize m-bit integers using a quantum circuit of size (i.e., number of gates) $O(m^2)$.

[18] has recently further improved the quantum space efficiency of Regev's algorithm while keeping the circuit size the same. They managed to constructs a quantum factoring circuit using $O(m \log m)$ qubits and $O(m^{3/2} \log m)$ gates. While Shor's factoring algorithm has always assumed that it operates on noiseless quantum device

Since a true error free qubit is sacred, a lot of effort has been focused on constructing fewer qubits to factor an m-bit number. [27] has reported a universal quantum algorithm for integer factorization by combining the classical lattice reduction with a quantum approximate optimization algorithm (QAOA). The number of qubits required is $O(\frac{m}{\log m})$, which is making it the most qubit-saving factorization algorithm to date. The authors also demonstrate the algorithm experimentally by factoring integers up to 48 bits with 10 superconducting qubits. They project that a quantum circuit under current noisy quantum computerswith 372 physical qubits and a depth of thousands might be able to factor 2048-bit RSA.

Although quantum theory has existed for an entire century, there is still a quest on how this extension on a probabilistic factoring algorithm affects crytographic computation.

Specifically, identifying and demonstrating a factoring problem is uniquely solvable with quantum technology remains an open concern. [8].

Nevertheless, most algorithms being used so far are based on searching a period of a generator a modulo N. A simple algorithm can be designed to factor N. In general, a period is as large as N itself. A new algorithm is called for here. There are few expensive elements to avoid here.

i. Computing a large period.

ii. Computing power mode operation which is costly.

iii. Utilising a complex field which can be hardly tuned and improved upon.

Thus, an efficient implementations of Shor's Algorithm has been progressively explored in the last 3 decades to reduce the required number of qubits and/or gate counts.

2 RSA cryptosystem

R. L. Rivest, A. Shamir, and L. Adleman have invented a technique for public key cryptography in 1977 [20], which came to be known as RSA cryptosystem.

A key generation process of RSA cryptosystem is given as follows;

1. Generate n-bits primes P and Q.
2. Compute the modulus $N = P \cdot Q$.
3. Set the public exponent $e = 2^{16} + 1$.
4. Compute private exponent $d = e^{-1} \pmod{(P-1)\cdot(Q-1)}$.
5. Set Public key (e, N) and Private key (d,N).

RSA is the most popular PKI due to the following reasons;

1. It is the first PKI
2. It is written in a simple formula.
3. It follows few thousand years' concept of prime numbers.
4. It is being written in cryptographic textbooks and taught in classes.
5. It is the standard to reckon with.

There are many difficult mathematical problem within RSA cryptosystem. Basic mathematical problems listed as follows;

1. Integer Factorization problem. Given N, it is difficult to factor P and Q out of $N = PQ$.
2. It is difficult to obtain a private key (d, N) from a public key (e, N).
3. It is difficult to compute $\phi(N) = (P-1)(Q-1)$.
4. The discrete log problem: For a selected plaintext $m = C^d \pmod{N}$. It is difficult to compute private exponent d.
5. The root problem: Given a ciphertext $C = M^e \pmod{N}$. It is difficult to compute the plaintext m.

In fact, any computational process which leads to solving an integer factoring is considered a difficult mathematical problem.

3 Shor's Algorithm

Peter Williston Shor from MIT has devised a quantum algorithm for factoring N which will runs in polynomial time [22]. Given an RSA modulo N, the Shor's Algorithm works as follows:

1. Choose a number a, which is relatively prime to N.
2. Find the period of the function $f(x) = a^x \pmod N$. This effort reduces to finding $f(\phi) \equiv 1 \pmod N$, where ϕ is a multiple of period of $f(\phi)$.
3. if $f(\phi)$ is even or $a^{\phi/2}$ has an integer value, then proceed further. Otherwise, choose another a.
4. Once the period ϕ is found, factors of N can be extracted from $\gcd(a^{\phi/2} \pm 1, N)$.

All computations can be performed classically in polynomial time but finding the period ϕ. For a small N, a period ϕ can be approximated by $N - 2\lfloor\sqrt{N}\rfloor$. Here comes a quantum computer role in play. A quantum phase search will carried out by an inverse QFT. This search allows for the period ϕ to be found in polynomial time.

Shor's Algorithm relies heavily on modular arithmetic, quantum parallelism and QFT. Theoretically, Shor's Algorithm is claimed to be capable of factoring a large RSA modulo N via a quantum computer in polynomial time.

4 An Overview on Classic Factoring Periods

Analogously, searching for a period within a ring modulo N has been a classic RSA modulo factoring strategy during RSA early years for 2 decades prior to the year 2000.

Table 1.

Table 1: A periodic ring on RSA modulo factoring

Ring Curve	Period
Power Modulo	$(P-1)(Q-1)$
Lucas Sequences	$(P \pm 1)(Q \pm 1)$
Elliptic Curves	$(P + t_P + 1)(Q + t_Q + 1)$
Pell Curve Sequences	$(P \pm 1)(P^2 + P + 1))^{\epsilon_P}(Q \pm 1)(Q^2 + Q + 1)^{\epsilon_Q}$

Several rings with their possible periods have been listed in Table 1. Traditionally, an attack focused on a hidden ring modulo a smaller prime P. First, the power mod function $f(\mathrm{x}) = a^x \pmod N$ is a classic starting point on factoring exercises which carry a period $\phi = (P-1)(Q-1)$. Second, then come a popular Pollard's rho algorithm [17] which runs in $O(\sqrt{P})$ proportional to a square root of the smaller prime P. Not long after a period on the left $P-1$ of a prime P was manifested and strongly attacked, an algorithm to attack a new period on the right P+1 appeared [25] given by Lucas Sequences. These traditional factoring algorithms have been heavily guarded by strong prime criteria.

In order to avoid having smaller smooth periods and maintain the strenght against any attacks on both sides, criteria of strong primes were mandated. In 1985, an elliptic curve has made its way into cryptography. A basic curve $e : y^2 = x^3 + ax + b \pmod P$ is periodically varied at ${}^{\#}E = P + t_P + 1$ where $|t_p| < 2\sqrt{P}$. Lenstra introduced an elliptic-curve factorization method (ECM) which gets around this strong prime obstacle by considering a group of a random elliptic curve over the finite field $\mathbb{Z}_P$ [16]. An ECM algorithm had successfully managed to exploit a random smooth period. Since then strong

prime criteria had been rendered obselete. Consequently, an RSA modulo bitsize has been doubled and protected by its sheer size.

Recently, another periodic Pell sequence has been discovered which another period $P^2 + P + 1$ besides $P \pm 1$ [1]. This new additional period has not been fully explored on any weakness though doubling its bitsize.

5 A Bloch Sphere

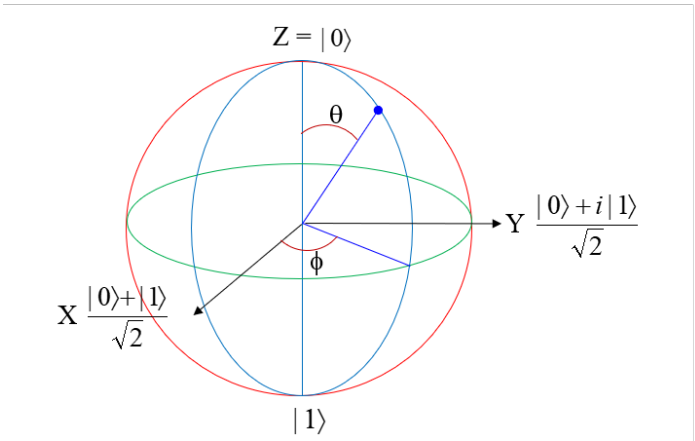


Figure 1: A qubit on a Bloch sphere in terms of polar coordinates.

Let a qubit $|q\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \alpha\begin{bmatrix}1\\0\end{bmatrix} + \beta\begin{bmatrix}0\\1\end{bmatrix}$ where $\alpha, \beta \in \mathbb{C}$ are complex numbers. Or alternatively as $|q\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$ Parameters θ and ϕ can be interpreted as spherical co-ordinates with magnitude one. A single qubit state can be plotted on the surface of a Bloch sphere as shown in Figure 1. For a given braket $\langle 0\rangle$, $\langle 0|$ is bra and $|0\rangle$ is ket.

6 Quantum Parallelism

Quantum parallelism refers to the ability of quantum computers to evaluate a function for multiple input values simultaneously. This capability can be achieved by preparing a quantum system in a superposition of input states, and applying a unitary transformation that encodes a function to evaluate. The resulting state encodes the function's output values for all input values in the superposition, allowing for the computation of multiple outputs simultaneously. This property is key to the speedup of many quantum algorithms.

Quantum algorithms that offer more than a polynomial speedup over the best-known classical algorithm include Shor's algorithm for factoring and the related quantum algorithms for computing discrete logarithms and solving Pell's equation. No mathematical proof has been found that shows that an equally fast classical algorithm cannot be discovered, although this is considered unlikely. Identifying cryptographic systems that may be secure against quantum algorithms is an actively researched topic under the field of post-quantum cryptography.

In 2021 at a quantum computing summit, IBM presented a 127-qubit microprocessor named IBM Eagle. Since then IBM has steadily scaling up its general-purpose quantum processor qubit size [5] as listed in Table 2.

Table 2: An achievable growth of IBM quantum computers.

Year	Name	Qubits
2021	Eagle	127
2022	Osprey	433
2023	Condor	1121
2024	Flamingo	1386
2025	Kookaburra	4158

7 An Open Question

There is an algorithm Shor's Algorithm which can factor $N = PQ$ and there is supposedly a growing quantum computer as shown in Table 2, why there is hardly a progress in factoring an RSA modulo in terms of bit security levels via a quantum computer?

Let us review some historical achievement of Shor Algorithm. Since an advent of Shor Algorithm, some people has only shown to factor small $N = PQ$ which typically carry 1-bit security level only. The list of RSA modulo N are presented in Table 3.

Table 3: A popular list of small integer factoring via a quantum computer.

Year	E	R	P	A	Q	N
2012	2	4	7	8	11	77
2014	2	8	233	236	241	56153
2018	2	10	2017	2021	2027	4088459
2020	2	12	1048589	1048594	1048601	1099551473989

Let $N = PQ$. Take $A = \lfloor \sqrt{N} \rfloor$ as a square root of N and R is the difference between the two primes P and Q. Let an exponent $e = P+Q-2A$. As the cycle $\phi = (P-1)(Q-1)$, then $N + 1 - 2a = \phi + E$. In general, it is intractable to solve discrete log problem on an exponent E. Choose a small integer a and take $a^{N+1-2A} = a^{\phi+E} \equiv a^E \pmod{N}$.

As the bit size of N grows, primes P and Q can be chosen such that e remains small to give an impression that a quantum computer can factor large RSA modulo N. These exercises have misled its readers by giving a wrong impression such a large N can be factored easily while in fact such an N can already be factored easily by a simple algorithm using a current computer in an instance.

Beyond a quantum lecture on factoring $3{\cdot}5 = 15$, a simple factoring effort has successfully factor 77 in 2012. Shor Algorithm concentrates on finding a cycle $\phi(N) = (P-1)(Q-1)$. In fact, finding a cycle within a periodic ring modulo N will also lead to factoring N. Let us take the first popular example. From [7], the largest number factored by a quantum computer is $56153 = 241 \cdot 233$. or

$$(N+1) - 2A = (P-1)(Q-1) + e = 56154 - 2 \cdot 236 = 55682$$

which gives $e = 2$. Next by [6], the largest number factored by a quantum computer is $4088459 = 2017 \cdot 2027$. Since $A = 2021$,

$$N + 1 - 2 \cdot a = 4084418.$$

But $(P-1)(Q-1) = 4084416$.

Lastly, in 2019 the largest number factored by [15] is $1099551473989 = 1048589 \cdot 1048601$ which appears to be large. Nevertheless, $a = 1048594$ and $(N+1)-2a = 1099549376802$.But $(P-1)(Q-1) = 10995512642800$. Therefore $e = 2$ only. From Table 3, a prime difference $R = Q - P$ is incrementing but $E = 2$ stay the same carrying security level 1-bit only.

8 A Typical RSA modulo

Practically, the Shor Algorithm need to retrieve the true period with very high probability under error control environment. An experimental exercise should be conducted on a steady growth of RSA modulo and its basic security level. Let us take the first prime P as the next prime of n-bit number from constant e minus 2 and the second prime Q as the next prime of n-bit number from $\pi^2 minus 9$. The list prime samples and their product are listed in Table 4 below. An exponent $e = P + Q - 2\lfloor\sqrt{2}\rfloor$ is also given for each N. The bit size n_E is considered a basic security level of respective RSA modulo N. It should be observed that initially e grows gradually. Traditional primes P and Q are closer to each other and E is just several bit smaller.

For that purposes, we have explore the implementation and performance of Shor's algorithm using Qiskit's quantum package on IBM Quantum hardware. IBM Quantum hardware refers to the physical quantum computing devices developed and maintained by IBM. These devices leverage the principles of quantum mechanics to perform computations that would be challenging or impossible for classical computers. IBM Quantum hardware is part of IBM's broader quantum computing initiative, which includes both hardware and software to advance the field of quantum computing.

Table 4: An increasing pseudo random RSA modulo.

n	P	Q	$\lfloor\sqrt{N}\rfloor$	N	E	n_E	time(s)
4	13	17	14	221	2	1	0.0
6	47	59	52	2773	2	1	0.0
8	191	223	206	42593	2	1	0.0
10	739	907	818	670273	10	4	0.0
12	2953	3571	3247	10545163	30	5	0.0
14	11777	14249	12954	167810473	118	7	0.0
16	47087	56993	51803	2683629391	474	9	0.01
18	188299	227977	207190	42927841123	1896	11	0.02
20	753187	911851	828730	686794319137	7578	13	0.10
22	3012719	3647393	3314901	10988570191567	30310	15	0.14
24	12050777	14589559	13259544	175815522037343	121248	17	0.54
26	48203081	58358173	53038134	2813043740131013	484986	19	2.18
28	192812311	233432701	212152536	45008698542782011	1939940	21	8.81
30	771249257	933730619	848610067	720139046141900083	7759742	23	34.03
32	3084996979	3734922479	3394440243	11522224564514190941	31038972	25	352.51

Let us take the case $N = 8$ where $(P, Q, N) = (191, 223, 42593)$ and go through a basic Shor's Algorithm. A median $A = \lfloor\sqrt{N}\rfloor = 206$. First, choose a small generator a where $1 < a < A$ and $\gcd(a, N) = 1$. Take $a = 3$ and Shor's Algorithm is supposedly able to approximate a period r such that $a^r \equiv 1 \pmod{N}$.

Once $r = \frac{(P-1)(Q-1)}{2} = 21090$ is reached, then $\gcd(a^{\frac{r}{2}} - 1, N)$ and $\gcd(a^{\frac{r}{2}} + 1, N)$ can be computed which have a good chance of containing P and Q. In this case, $a^{\frac{r}{2}} = 2675$. Then $a^{\frac{r}{2}} - 1 = 2674 = 2 \cdot 7 \cdot 191$ and $a^{\frac{r}{2}} + 1 = 2676 = 2^2 \cdot 3 \cdot 223$. Consequently, factors P and Q of N can be numerically extracted.

9 Quantum Fourier Transform (QFT)

Shor's Algorithm focuses on a power modulo periodic functions. This periodic search has been proclaimed to produce a real-time quantum solution for factoring $2n$-bit RSA modulo with time complexity $O(n^3)$.

There is a simple rule for measurement. To find the probability of measuring a state $|y\rangle$ in the qubit $|x\rangle$, compute a probability;

$$P(|x\rangle) = |\langle x|y\rangle|^2.$$

The symbols $\langle$ and $|$ tell us $\langle x|$ is a row vector and the symbols $|$ and $\rangle$ tell us $|y\rangle$ is a column vector. In quantum mechanics, column and row vectors are referred to as a ket and the bra respectively. Together they make up a bra-ket notation. Any ket $|a\rangle$ has a corresponding bra $\langle a|$ via its conjugate transpose.

Similarly, a quantum Fourier transform is an analogue of a discrete Fourier transform and acts on a quantum state vector $|x\rangle = \sum_{j=0}^{N-1} x_j|j\rangle$ and maps it to the quantum state vector,

$$|y\rangle = \sum_{k=0}^{N-1} y_k|k\rangle$$

via a formula,

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w_N^{jk}$$

where $w_N^{jk} = e^{2\pi i \frac{jk}{N}}$.

A QFT can be performed efficiently on a quantum computer with a decomposition into the product of simpler unitary matrices. This transform can also be expressed as the map:

$$|j\rangle :\to \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} w_N^{jk}|k\rangle$$

And in terms of the unitary matrix:

$$\mathrm{U_{QFT}} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} w_N^{jk}|k\rangle\langle j|$$

Let us derive a QFT acting on an m-qubit. Let a state $|x\rangle = |x_{m-1}\cdots x_1x_0\rangle$ written in little endian and $w_N^{xy} = e^{2\pi i \frac{xy}{N}}$;

$$\begin{aligned}
\mathrm{U_{QFT}}|x\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} w_N^{xy}|y\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i \frac{xy}{N}}|y\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{(2\pi i x \sum_{k=0}^{m-1} \frac{y_k}{2^k})}|y_{m-1}\cdots y_1y_0\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \prod_{k=0}^{m-1} e^{2\pi i x \frac{y_k}{2^k}}|y_{m-1}\cdots y_1y_0\rangle \\
&= \frac{1}{\sqrt{N}} \bigotimes_{k=0}^{m-1} (|0\rangle + e^{2\pi i \frac{x}{2^k}}|1\rangle) \\
&= \frac{1}{\sqrt{N}} \otimes (|0\rangle + e^{2\pi i \frac{x}{2^{m-1}}}|1\rangle) \otimes \cdots \otimes (|0\rangle + e^{2\pi i \frac{x}{2^i}}|1\rangle) \otimes (|0\rangle + e^{2\pi i \frac{x}{2^0}}|1\rangle).
\end{aligned}$$

In a classical computer, at any instances, an input $\mathbf{x} = x_{m-1}\cdots x_1x_0$ can be efficiently transformed via a fast fourier transform into an output $\mathbf{y} = y_{m-1}\cdots y_1y_0$. In a quantum computing, a state $|x\rangle = |x_{m-1}\cdots x_1x_0\rangle$ can also be presumably transformed into a quantum state $|y\rangle = |y_{m-1}\cdots y_1y_0\rangle$.

The best QFT algorithm known (as of late 2000) require only $O(n \log n)$ gates to achieve an efficient approximation [14]. For correctness, let us take a look at a compact circuit for a 3-qubit QFT in Figure 3 as follows:

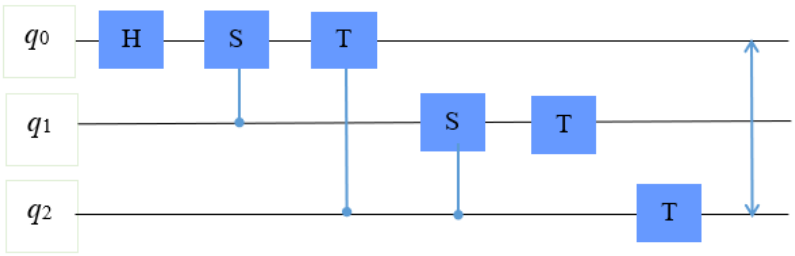


Figure 2: A raw 3-qubit quantum circuit on Quantum Fourier Transform.

S and T gates are phase and $\pi/8$ gates. A matrix of QFT may be written explicitly as

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^6 & w^7 \\ 1 & w^2 & w^4 & w^6 & 1 & w^2 & w^4 & w^6 \\ 1 & w^3 & w^6 & w & w^4 & w^7 & w^2 & w^5 \\ 1 & w^4 & 1 & w^4 & 1 & w^4 & 1 & w^4 \\ 1 & w^5 & w^2 & w^7 & w^4 & w & w^6 & w^3 \\ 1 & w^6 & w^4 & w^2 & 1 & w^6 & w^4 & w^2 \\ 1 & w^7 & w^6 & w^5 & w^4 & w^3 & w^2 & w \end{bmatrix}$$

where w is an 8th primitive root of unity. A 3-qubit QFT here is infact an 8 qubit state vector.

QFT is a quantum implementation of the discrete Fourier transform. Let us carry out QFT algorithm to transform M qubit state vector;

$$|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_M|M\rangle$$

to its Fourier transform

$$|\beta\rangle = \beta_0|0\rangle + \beta_1|1\rangle + \cdots + \beta_M|M\rangle.$$

A measurement on $|\beta\rangle$ will be referred to as a quantum Fourier sampling. Let w be an M-th primitive root of unity, $w = e^{2\pi/M}$. Then, discrete QFT is defined by

$$\mathrm{QFT}_M = \frac{1}{\sqrt{M}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & w & w^2 & w^3 & w^4 & w^5 & w^{M-2} & w^{M-1} \\ 1 & w^2 & w^4 & w^6 & w^8 & w^{10} & w^{2(M-2)} & w^{2(M-1)} \\ 1 & w^3 & w^6 & w^9 & w^{12} & w^{15} & w^{3(M-2)} & w^{3(M-1)} \\ 1 & w^4 & w^8 & w^{12} & w^{16} & w^{20} & w^{4(M-2)} & w^{4(M-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & w^{M-2} & w^{(M-2)2} & w^{(M-2)3} & w^{(M-2)4} & \cdots & w^{(M-2)(M-2)} & w^{(M-2)(M-1)} \\ 1 & w^{M-1} & w^{(M-1)2} & w^{(M-1)3} & w^{(M-1)4} & \cdots & w^{(M-1)(M-2)} & w^{(M-1)(M-1)} \end{bmatrix}$$

While a promise of quantum physical optimizers with having more than 1000 qubits offers interesting perspectives, the tests carried out in many fields do not yet produce any consequential results about quantum factoring supremacy [10]. In quantum computation, an m-qubit QFT is defined as a 2^m-point DFT on a quantum superposition of the signal of length 2^m. A quantum convolution is conjectured to be physically impossible [12]. First practical question here, does QFT_M can be internally computed at an instance all the time in an instance following the sheer size of M? M will carry the same size of N.

Second practical question here, can a quantum computer take a superposition of Hadamard gate Unitary gate $H^{\otimes m}$ internally computed at an instance following the sheer size of M? M must carry the same size of N.

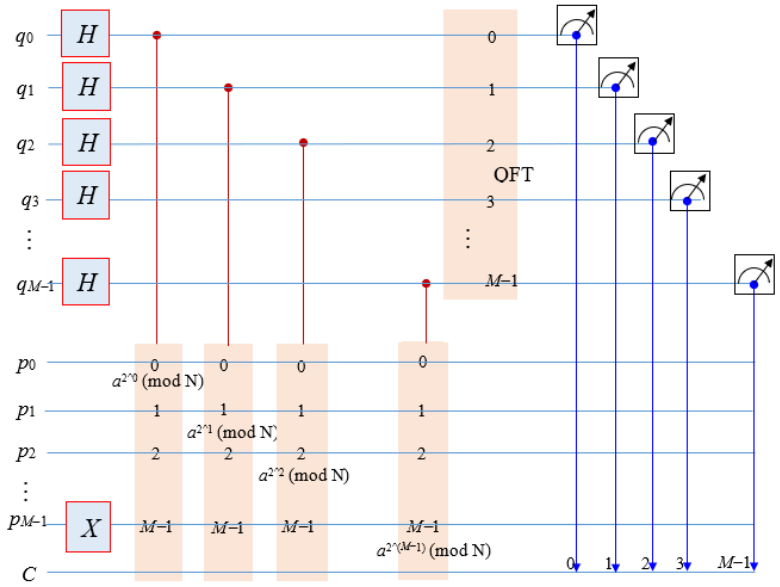


Figure 3: A raw quantum circuit on QFT Shor's Algorithm will come down to a period finding problem.

10 Power Modulo Function

A bulk of an execution time is spent on performing a modular exponentiation. Let start from a state x and transform it into state $xa \pmod N$.

$$U_{a,N}|x\rangle = |xa \pmod N\rangle.$$

Applying this gate one at a time,

$$\begin{aligned}
U^0_{a,N}|x\rangle &= |1 \pmod N\rangle \\
U^1_{a,N}|x\rangle &= |a \pmod N\rangle \\
U^2_{a,N}|x\rangle &= |a^2 \pmod N\rangle \\
U^3_{a,N}|x\rangle &= |a^3 \pmod N\rangle \\
&\vdots \\
U^r_{a,N}|x\rangle &= |a^r \pmod N\rangle.
\end{aligned}$$

a power modular operation will be called as many time as needed.

Start with a register of $m = 2N$ qubits and initialize them all in the zero state $|0\rangle$. We still need to prepare the register into a uniform superposition by applying Hardamard H gate to each qubit in the register

$$|\psi\rangle = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle$$

Since we cannot afford to build an m-qubit quantum circuit, we can make a m-qubit query at a time where $m << m$. However, m must be larger than $\sqrt{M}$ in order to gain

a constant probability of success. The lower bound $\sqrt{M}$ on a quantum search has been proven prior to the discovery of Grover's Algorithm.

A raw quantum circuit to compute this power modulo operation is depicted in Figure 2. Third practical question here, does Unitary gate $U_{a,N}$ can be internally computed at an instance all the time in an instance to period r following the sheer size of r? A period r will carry about the same size of N.

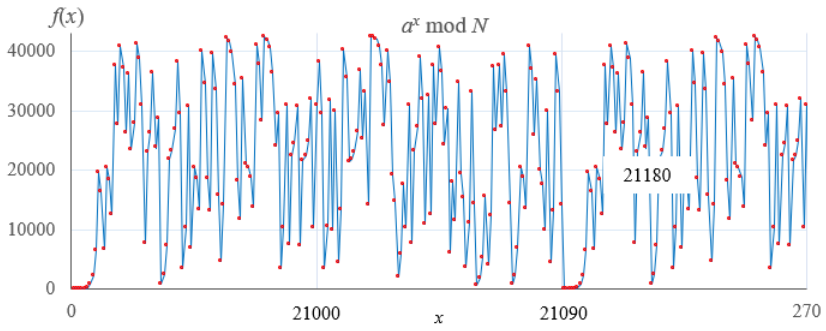


Figure 4: A truncated periodic signal on a power mod function.

11 A Period Finding Problem

Let $f : \{0, 1, 2, \ldots, m-1\} \to S$ such that $f(x) = f(x+r)$ for all x. The challenge here is to find a period r. We need to find a period r such that if $f(x) = a^x \pmod N$, then a collision is the case $f(x) = f(x+r)$. As depicted in Figure 3, there are hardly practical to assign M qubits on the left and measure M qubit on the right.

Figure 4 depicted a power mod function on $3^x \pmod{42593}$ periodic at $r = \frac{(P-1)(Q-1)}{2} = 21090$. Theoretically, a quantum computer can easily find a small period r via QFT. Without a period r, we can estimate it to be smaller than $\frac{N+1-2A}{2}$ where $N + 1 - 2a = (P-1)(Q-1) + E$ which is still a $2n$-bit number or several bit smaller.

In order to search for a period r, we need to locate a collision that suffices to check on $\sqrt{M} = \sqrt{\frac{N+1-2A}{2}}$ of n-bit cases. Searching for a collision has been identified as Birthday Paradox. In summary, there are 365 days per year. We need to randomly check on $\sqrt{365} \cong 19.105$ persons to gain a probability above $\frac{1}{2}$.

While r is at most $(2n-1)$-bit number, how many instances we need to check on in order to gain a collision with a constant probability of success. The answer is $\sqrt{r}$ which is an n-bit number. There is another equivalent unsorted search algorithm, namely, Grover's Algorithm which must run in $O(\sqrt{M})$ which already reaches the lower search via quantum computers.

12 Grover Theorem

In 1996, Lov Kumar Grover proposed a quantum algorithm to find a given element in an unstructured of m elements in $O(\sqrt{M})$ queries compared to $\mathrm{O}(M)$ required classically [13]. Grover's algorithm could theoretically brute-force current 128-bit symmetric AES cryptographic key in 2^{64} iterations.

Given an unstructured function f,

$$f : \{0, 1, 2, \ldots, M-1\} \to \{0, 1\},$$

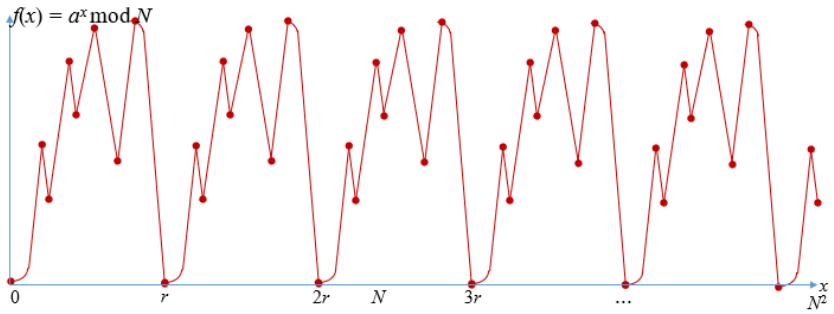


Figure 5: A wide window on an extended periodic signal on a power mod function

Any searching quantum algorithm will take at least $\sqrt{M}$ steps.

A proof of this Theorem 1 is beyond the scope of this paper. Suppose there are M possible indices to locate $f(x) = a^x \equiv 1 \pmod{N}$ items for the search problem and they are indexed by assigning each item an integer $0, 1, 2, \ldots, M-1$. Assuming a power mod function f is nor structured, $\sqrt{M}$ measures are required to gain a successful search. Fourth practical question here, does Shor's Algorithm breach the lower bound on quantum search. One can argue here classically the function f is practically unstructured on domain set $\frac{M}{\log M}$.

13 Quantum Period Finding via Quantum Phase Estimation

Let us extend a domain window to the right up to N^2 as shown in Figure 5. This idea appears to be useful in capturing several periodic indices. The real problem here to get anywhere near a period. Extending to a wider window will hardly increase a chance to capture 2 periodic points.

Suppose f is periodic with period r, Then a QFT of f, namely, $\hat{f}$ is periodic with period $\frac{M}{r}$. Currently, the Shor Algorithm can be easily employed to factor 32-bit strong RSA modulo by error-free quantum computing simulation. Take P and Q as $n = 16$-bit primes. $N = PQ$ is 32-bit integer. In order to quantum registers of this case, we need a 2^{32} vector to form about 4 billion registers. Then we need to do QFT on these billions of registers. Here, we start to enter a huge memory issue.

In order to be practical, a number of qubits must be limited the number of classic computing bits. Let initialize m qubits as $|q_{m-1} \ldots q_2 q_1 q_0\rangle = |0 \ldots 000\rangle$ and $|\lambda_{m-2} \ldots \lambda_2 \lambda_1 \lambda_0\rangle = |0 \ldots 001\rangle$. Take m as the bitsize of N, in this paper, $m = 2N$. Traditionally, a target period is slightly smaller than N, the second set of qubit register is set as $m - 2 = 2N - 2$. In this setting, [9] argues that theoretically a probability of success in getting a true periodic phase is approximately one.

14 Current Status on Shor's Algorithm

Next year, there will be 2 possibles cases on an m-bit RSA modulo. Take a sample RSA1024 from RSA challenge numbers. A critical quest here is 'Can a quantum computer practically break RSA1024 modulo next year?'

Case 1: By looking at the growth of IBM quantum computers in Table 2, there will be sufficient number of qubits, specifically $2(1024) + 6$ qubits to break RSA1024. These quantum machine are presumably succumb to a minimum acceptable error.

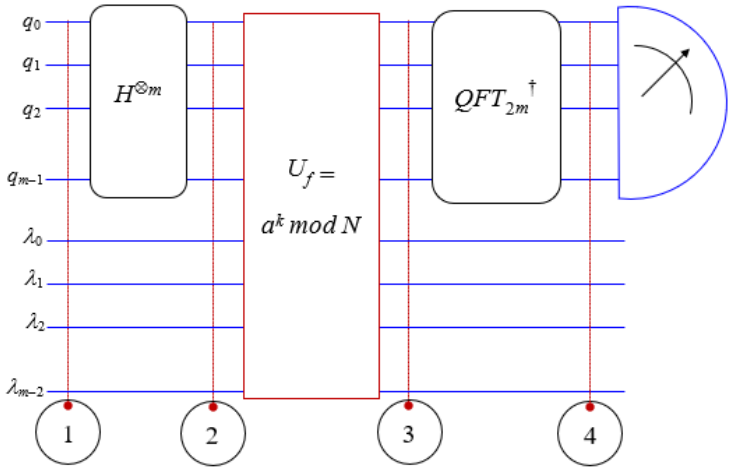


Figure 6: Quantum order finding via a phase estimation

Case 2: There will be $(2m + 6)$-qubit quantum computers with a minimum acceptable error. However, Shor Algorithm is still hardly practically viable to factor a 128-bit RSA modulo.

This paper will argue that the latter case is favourable. Hence, there is a need to have another algorithm to practically factor via a quantum computer. This position will certainly contradict a common belief on Shor Algorithm specifically by [9] on the success probability of quantum order finding.

Moreover, given a quantum computer in few thousand qubits as listed in Table 2, RSA1024 should be factorable by next year. The first hurdle is to overcome factoring an RSA modulo which carry security bit higher than 64 bits following a typical growth in Table 4.

Currently, there are 4 major practical hurdles to execute a factoring Shor Algorithm via a phase estimation at 4 various locations in Figure 5.

1) A parallel definition of a sequence of m-qubits must be initially or internally declared. This declaration will take up a superposition at 2^m spaces.

2) Unitary gate $U_{a,N}$ must be able to internally compute a power modulo operation in an instance. A power modulo operation has been replaced internally into multiply modulo operation one at a time.

3) Shor Algorithm implicitly calls for a QFT or QFT inverse algorithm to be computed in superposition using 2^m logical qubits.

4) A composition of the above 2 functions are extremely heavy in a superposition. Practically, a measurement must be finally made on an exponential internal composition.

At the same time, a quantum computer would manage to accurately estimate a period of power modulo function faster than $O(\sqrt{N})$ as stipulated in Grover Theorem.

15 A Special Case

A special attention should be given on an effort using the smallest number of qubit need to factor. (Yan and et. al, 2022) proposed a universal quantum algorithm for integer factorization that requires only sublinear quantum resources. The algorithm is based on

the classical Schnorr's algorithm [21] which uses lattice reduction to factor an RSA modulo N. For an m-bit integer N, the number of qubits needed for their algorithm is $O(\frac{m}{\log m})$ which is sublinear in the bit length of N. The authors reported that they have successfully factorized RSA modulo 1961, 48567227 and 261980999226229 using 3, 5 and 10 qubits in a superconducting quantum processor, respectively. An easy estimate would be 372 physical qubits is sufficient to factor an RSA-2048. Nevertheless, the number of gates is hardly projected in terms thousands if not billions.

Table 5: A list of small integer factoring via a quantum computer by sublinear qubits.

n	P	Q	A	N	E	n_E
6	37	53	44	1961	2	2
13	6133	7919	6969	48567227	114	7
24	15538213	16860433	16185827	261980999226229	26992	15

The RSA modulos being used here are well chosen in an increasing security n_E bits as shown in Table 5. This experimental result did not go beyond 15-bit security level. Might it be due to their simulation constraint or practical quantum constraint?

16 Critical Discussion

Currently, a quantum state qubit is operating on a minute error. Taking a larger sequence of m-qubits will accumulate the total error which is still under serious research. Running Shor's algorithm on a current fault-tolerant quantum computer is quite resource intensive to manage.

Are we going to see a quantum machine to factor 896 to 1024-bit RSA modulo next year 2025? Or the Shor algorithm is still crawling at factoring from 32 to 64-bit RSA modulo with 31 and 63-bit security levels respectively. A super computer can simulate this experiment. An Osprey IBM machine should do the job. However, the next challenge is on 128-bit RSA modulo which a supercomputer will reach its limit to simulate. Chances are everyone will blame it on quantum errors.

A number field sieve is an improvement to a quadratic sieve. A variation of quadratice sieve is the most efficient factoring algorithm in this arena. Using such an algorithm to factor a large number N, a running time of the number field sieve is super-polynomial. Chances are a superposition will give an advantage in searching for a candidate x such that $x^2 \equiv b \pmod N$ for some small integer b.

17 Conclusion

After 3 decades of its inception, Shor's Algorithm is still crawling at its infant stage. By now any factoring enthusiasts would have realized that the Shor algorithm will not practically perform better than classic periodic factoring algorithms. A practical analysis on this its development suggest that an exponential number of logical qubits is needed to push through Shor's Algorithm in practice. It has already reached the lower bound and exhausted its quantum super position prowess. Perhaps, a new algorithm is called for which can harness a quantum superposition advantages in a quadratic form.

References

[1] Nur Azman Abu and Abderrahmane Nitaj. A Cubic Pell Cryptosystem CP256-1299. In *Proceedings of the 8th International Cryptology and Information Security Conference*, pages 109–127, July 2022.

[2] R Alves, D Hildenbrand, J Hrdina, and C Lavor. An online calculator for quantum computing operations based on geometric algebra. *Advances in Applied Clifford Algebras*, 32:1–20, 2022.

[3] Stephane Beauregard. Circuit for Shor's algorithm using $2n + 3$ qubits. *Quantum Information and Computation*, 3(2):175–185, 2003.

[4] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. Reducing the Number of Qubits in Quantum Factoring. *Cryptology ePrint Archive*, 2024.

[5] C Choi. An IBM quantum computer will soon pass the 1,000-qubit mark. *IEEE Spectrum*, 24, 2022.

[6] Avinash Dash, Deepankar Sarmah, Bikash K Behera, and Prasanta K Panigrahi. Exact search algorithm to factorize large biprimes and a triprime on IBM quantum computer. *arXiv preprint arXiv:1805.10478*, 2018.

[7] Nikesh S Dattani and Nathaniel Bryans. Quantum factorization of 56153 with only 4 qubits. *arXiv preprint arXiv:1411.6758*, 2014.

[8] Casey Dowdle and James Whitfield. A Practical Introduction to Quantum Computing. *SIAM News*, May 2024, 2024.

[9] Martin Ekerå. On the success probability of quantum order finding. *arXiv preprint arXiv:2201.07791*, 2022.

[10] Gérard Fleury and Philippe Lacomme. A technical note for a Shor's algorithm by phase estimation. *arXiv preprint arXiv:2206.00757*, 2022.

[11] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[12] Artyom M Grigoryan and Sos S Agaian. 3-Qubit Circular Quantum Convolution Computation using Fourier Transform with Illustrative Examples. *arXiv preprint arXiv:2205.05727*, 2022.

[13] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[14] Lisa Hales and Sean Hallgren. An improved quantum Fourier transform algorithm and applications. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 515–525. IEEE, 2000.

[15] Amir H Karamlou, William A Simon, Amara Katabarwa, Travis L Scholten, Borja Peropadre, and Yudong Cao. Analyzing the performance of variational quantum factoring on a superconducting quantum processor. *npj Quantum Information*, 7(1):156, 2021.

[16] Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[17] J. M. Pollard. A Monte Carlo method for factorization. *BIT Numerical Mathematics*, 15(3):331–334, 1975.

[18] Seyoon Ragavan and Vinod Vaikuntanathan. Space-Efficient and Noise-Robust Quantum Factoring. *arXiv preprint arXiv:2310.00899*, 2024.

[19] Oded Regev. An Efficient Quantum Factoring Algorithm. *arXiv preprint arXiv:2308.06572*, 2024.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[21] Claus-Peter Schnorr. Fast Factoring Integers by SVP Algorithms. In Marc Fischlin and Stefan Katzenbeisser, editors, *Number Theory and Cryptography*, Lecture Notes in Computer Science (LNCS), pages 73–93. Springer, 2013.

[22] Peter W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[23] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum Networks for Elementary Arithmetic Operations. *Physical Review A*, 54:147–153, 1996.

[24] S. Whitlock and T. D. Kieu. Quantum Factoring Algorithm using Grover Search. *arXiv preprint arXiv:2312.10054*, 2023.

[25] H. C. Williams. A $p+1$ Method of Factoring. *Mathematics of Computation*, 39(159):225–234, 1982.

[26] Hayata Yamasaki and Masato Koashi. Time-Efficient Constant-Space-Overhead Fault-Tolerant Quantum Computation. *Nature Physics*, 2024.

[27] Bao Yan, Ziqi Tan, Shijie Wei, Haocong Jiang, Weilong Wang, Hong Wang, Lan Luo, Qianheng Duan, Yiting Liu, Wenhao Shi, et al. Factoring Integers with Sublinear Resources on a Superconducting Quantum Processor. *arXiv preprint arXiv:2212.12372*, 2022.

[28] Yewei Yuan, Chao Wang, Bei Wang, Zhao-Yun Chen, Meng-Han Dou, Yu-Chun Wu, and Guo-Ping Guo. An Improved QFT-Based Quantum Comparator and Extended Modular Arithmetic Using One Ancilla Qubit. *New Journal of Physics*, 25:103011, 2023.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 64–74.

Exposing Vulnerabilities in a Post-Quantum Implicit Certificate Scheme

Tao-hsiang Chang[1,2], Jen-Chieh Hsu[1], Raylin Tso[1] and Hao-Yi Hsu[1]

[1] National Chengchi University, Taipei, Taiwan
tchang@cs.nccu.edu.tw 105753501@nccu.edu.tw
raylin@cs.nccu.edu.tw; 112753501@g.nccu.edu.tw
[2] Kanazawa University, Kanazawa, Japan

Abstract. The Post-Quantum Cryptography McEliece-Chen (PQCMC) scheme, designed to withstand quantum computational threats, is critically examined in this paper. Our analysis unveils a significant flaw within the scheme, where an end entity (EE) can potentially derive the Certificate Authority's (CA) private signing key, thereby compromising the system's integrity and authenticity. This vulnerability stems from the scheme's reliance on invertible matrices and the processes surrounding them, which we demonstrate could lead to the unauthorized creation of valid certificates. Such a breach not only questions the robustness of PQCMC against conventional attacks but also its adequacy for post-quantum security in practice. Our findings emphasize the urgent need for rigorous, dynamic security evaluations in developing cryptographic protocols, ensuring they are secure against both quantum and sophisticated conventional threats.

Keywords: Post-Quantum Cryptography · McEliece Cryptography · Cryptographic Vulnerabilities · Implicit Certificates

1 Introduction

Public Key Infrastructure (PKI) [1, 34, 23] plays a crucial role in securing digital communications by providing mechanisms for secure key exchange, authentication, and encryption. Central to PKI are digital certificates [18, 24], which associate public keys with entities, enabling verification of the entities' credentials. There are generally two types of certificates: explicit [8, 12], which contain signatures from a Certificate Authority (CA), and implicit, which do not require such signatures but use other methods for entity authentication.

Implicit certificates [9, 6] differ fundamentally from their explicit counterparts by minimizing the role of the CA in the certification process. While explicit certificates involve the CA directly signing the certificate to attest to the authenticity of the public key, implicit certificates use a clever combination of key agreement protocols [5, 16] and partial key escrow [4, 3] to authenticate the public key. In this model, the CA contributes part of the public key as a commitment during certificate issuance, which allows the certificate holder to derive the complete public key in a manner that can be independently verified by third parties without direct CA involvement. This approach not only reduces the computational load and complexity associated with CA operations but also enhances privacy and scalability by decreasing the CA's visibility into the subsequent use of the certificate.

The strength of implicit certificates lies in their streamlined process and reduced reliance on CA resources, making them particularly suitable for environments where computational resources are scarce or the risk of CA compromise is non-negligible. These certificates are especially advantageous in distributed systems [10, 31], Internet of Things (IoT)

environments [25, 30], and mobile applications [26], where the overhead associated with traditional certificate verification can be prohibitively expensive or technically challenging.

However, the advent of quantum computing [28, 15, 17] poses significant threats to contemporary cryptographic systems, including those underlying PKI. Quantum computers can efficiently solve problems such as integer factorization and discrete logarithms, which underpin many of today's cryptographic protocols, including RSA and Elliptic Curve Cryptography (ECC). This capability, derived primarily from algorithms like Shor's [27, 33], has spurred global initiatives to develop cryptographic systems resilient to both classical and quantum threats.

Among these initiatives is the Post-Quantum Cryptography (PQC) standardization [21] effort led by the National Institute of Standards and Technology (NIST). This effort aims to evaluate and standardize cryptographic algorithms that can ensure security in the era of quantum computing. While NIST's work primarily focuses on a broadly applicable suite of algorithms, including public-key encryption and digital signatures, it does not specifically cater to the unique requirements of certificate issuance, particularly implicit certificates.

In response to this gap, the Post-Quantum Cryptography McEliece-Chen (PQCMC) implicit certificate scheme [11] was developed. This scheme claims to be a robust candidate for PQC, leveraging the McEliece cryptography system [14, 29, 2], known for its resistance to quantum attacks due to the NP-hard problem of decoding linear codes [7, 19]. Despite its potential, our investigation reveals significant vulnerabilities within the PQCMC scheme that could allow an end entity (EE) to undermine the integrity of the certificate issuance process, enabling the unauthorized creation of certificates. This discovery raises serious concerns about the use of PQCMC in secure communications [20, 32, 22], particularly in scenarios where trust and authentication are paramount.

The objective of this paper is to detail our discovery of this critical flaw in the PQCMC scheme, which allows an EE to derive the CA's private signing key through algebraic manipulation of available information provided to the EE. We provide a demonstration of how this vulnerability can be exploited, enabling unauthorized certificate creation. Our analysis discusses the implications of this flaw for the security of digital communications and contributes to the ongoing development of secure post-quantum cryptographic protocols.

1.1 Organization

This rest of this paper is structured as follows: Section 2 discusses the foundational aspects of McEliece Cryptography. Section 3 details the PQCMC scheme, including its design and security properties. Section 4 analyzes a significant vulnerability within the PQCMC scheme, detailing the technical mechanism of the flaw, providing a conceptual illustration, and discussing its implications. Section 5 concludes the paper, summarizing our findings and their implications for post-quantum cryptography.

2 Preliminaries

This section provides the necessary background on McEliece cryptography, which forms the basis of the PQCMC scheme. Developed by Robert McEliece in 1978, the McEliece cryptography system is a prominent code-based cryptographic scheme. It is notable for its reliance on the NP-hard problem of decoding a general linear code. The system, originally employing Goppa codes, offers efficient encoding and decoding processes while maintaining robust security against known attacks. These characteristics make it particularly resistant to quantum computing threats, a key reason for its use in post-quantum cryptographic schemes like PQCMC [11].

2.1 McEliece Encryption Scheme

The McEliece system operates using a comprehensive set of keys, each playing a critical role in the encryption and decryption processes. These elements of keys are outlined in Table 1.

Table 1: Elements of Keys in McEliece Cryptography.

Key	Description	Dimensions	Feature
K_1	Scrambler Matrix	$\zeta_1 \times \zeta_1$	Invertible
K_2	Generator Matrix	$\zeta_1 \times \zeta_2$	$K_2K_4 = I$
K_3	Permutation Matrix	$\zeta_2 \times \zeta_2$	Invertible
K_4	Decoder Matrix	$\zeta_2 \times \zeta_1$	$K_4K_2 = I$
K_5	Error-Detector Matrix	Varied dimensions	Clears error

The public key, denoted as L, is derived from the product of K_1, K_2, and K_3, forming $L = K_1K_2K_3$. The secret key comprises the set $SK = \{K_1, K_2, K_3\}$, where each component is essential for the decryption process. Notably, K_4 and K_5 are generated based on K_2, playing critical roles in decoding and error correction, respectively. These keys facilitate the encryption and decryption operations, which are further detailed in Algorithms 1 and 2. To better integrate with the PQCMC scheme, the algorithms as well as notations here are mostly repeated with the ones introduced by Chen [11] with minor modifications.

Algorithm 1 Encryption Algorithm $z = Enc(\mathbf{m}, L)$

Require: Plaintext $\mathbf{m}$ of dimension $1 \times \zeta_1$, Public Key L of $\zeta_1 \times \zeta_2$
Ensure: Ciphertext $\mathbf{z}$ of $1 \times \zeta_2$
1: Generate random error vector $\mathbf{e}$ of dimension $1 \times \zeta_2$
2: $\mathbf{z} \leftarrow \mathbf{m}L + \mathbf{e}$ ▷ Ciphertext is the encoded message plus error
3: **return** $\mathbf{z}$

An error-correcting function $f(\alpha, K_5)$ is used in the decryption process to detect and correct errors introduced by the vector $\mathbf{e}$. The function f ensures that the received message $\mathbf{z}'$, once error-corrected, yields $\mathbf{m}'$, which represents the plaintext post-encoded with K_1K_2. This error correction is critical to recovering the original message from its encrypted form.

Algorithm 2 Decryption Algorithm $m = Dec(\mathbf{z}, SK)$

Require: Ciphertext $\mathbf{z}$ of dimension $1 \times \zeta_2$, Secret Key SK
Ensure: Plaintext $\mathbf{m}$ of dimension $1 \times \zeta_1$
1: Decompose SK and compute to obtain $K_1^{-1}, K_4, K_5, K_3^{-1}$ where the dimensions are listed in table 1
2: $\mathbf{z}' \leftarrow \mathbf{z}K_3^{-1}$ ▷ Remove permutation layer
3: $\mathbf{m}' \leftarrow f(\mathbf{z}', K_5)$ ▷ Apply error correction to retrieve encoded message
4: $\mathbf{m} \leftarrow \mathbf{m}'K_4K_1^{-1}$ ▷ Decode and remove scrambler
5: **return** $\mathbf{m}$

2.2 McEliece-based Digital Signatures in PQCMC

McEliece can also be adapted for digital signatures. The signing and verification process leveraging McEliece cryptography is described in Algorithms 3 and 4. It's important to note that this signing mechanism is specific to the PQCMC scheme [11] and differs from the well-recognized general McEliece-based signature scheme proposed by Courtois et

al. [13]. We believe this specificity contributes to the vulnerability discussed later in this paper.

Algorithm 3 Signing Algorithm $\mathbf{s} = Sign(\mathbf{m}, SK)$

Require: Message $\mathbf{m}$ of dimension $1 \times \zeta_1$, Secret Key SK
Ensure: Signature $\mathbf{s}$ of dimension $1 \times \zeta_2$
1: Decompose SK and compute to obtain K_1^{-1}, K_4, K_3^{-1} where the dimensions are listed in table 1
2: $\mathbf{s}^T \leftarrow K_3^{-1} K_4 K_1^{-1} \mathbf{m}^T$ ▷ Encode message $\mathbf{m}$ using secret key components
3: **return s**

Algorithm 4 Verification Algorithm $\mathbf{m} = Ver(\mathbf{s}, L)$

Require: Signature $\mathbf{s}$ of dimension $1 \times \zeta_2$, Public Key L of $\zeta_1 \times \zeta_2$
Ensure: Original message $\mathbf{m}$ of $1 \times \zeta_1$
1: $\mathbf{m}^T \leftarrow L\mathbf{s}^T$ ▷ Decode the signature $\mathbf{s}$ using the public key L to recover message $\mathbf{m}$
2: **return m**

In the signing and verification processes (Algorithms 3 and 4), the identity matrix I is leveraged that $K_3^{-1} K_4 K_1^{-1} L = I$. This identity is pivotal as it demonstrates that the operations involved in the signing algorithm effectively prepare a message $\mathbf{m}$ in a way that, when processed through the verification algorithm using the public key L, will return the original message $\mathbf{m}$. This ensures that the signature $\mathbf{s}$, when decoded by L, accurately reconstructs the signed message, affirming the signature's validity. Consequently, $SK_s = K_3^{-1} K_4 K_1^{-1}$ is defined as the transformation component of the secret key used in the signing process.

3 Post-Quantum Cryptography McEliece-Chen Scheme Overview

This section delves into the specifics of the PQCMC implicit certificate scheme [11], offering a detailed explanation of its workings and the security properties it aims to achieve. By understanding the mechanics and intended features of the PQCMC, we can better appreciate the significance of the vulnerability uncovered and its implications.

3.1 Generation of Random Invertible Matrices

The PQCMC scheme introduces a novel method for generating random invertible matrices. This method ensures the creation of M_r and M_h, based respectively on the random number r and the hash value h in the protocol. Here, we briefly describe this innovative matrix generation method and its application within the PQCMC scheme:

- **Seed Initialization:** Both matrices are generated by seeding a pseudorandom number generator with r for M_r and h for M_h. This seed ensures that each matrix is unique and securely tied to its originating data.

- **Matrix Filling and Verification:** The matrices are filled using the outputs from the pseudorandom number generator. The generation process includes checks to ensure that each matrix is invertible—a critical requirement for the matrices to function correctly within the cryptographic operations of PQCMC.

- **Algorithmic Efficiency:** The method is designed to be computationally efficient, enabling the quick generation of invertible matrices even in resource-constrained environments, which is vital for maintaining the overall efficiency of the certificate issuance process.

Notably, this matrix generation method, denoted as MGM($\cdot$), generates two matrices simultaneously. From a given random number r, it produces $(M_r, M_r^{-1}) \leftarrow \text{MGM}(r)$, where $M_r M_r^{-1} = I$. This method enhances both the security and efficiency of the PQCMC scheme, marking a significant advancement in cryptographic practices, particularly in preparing for the challenges posed by quantum computing.

3.2 Detailed Description

The PQCMC scheme leverages McEliece cryptography and the ECQV framework, aiming to securely issue implicit certificates in environments vulnerable to quantum attacks. The interaction between the EE and the CA is outlined in Figure 1 and involves several critical steps and components:

EE		CA
Generates $\{K_{1,(E)}, K_{2,(E)}, K_{3,(E)}\}$		
Generates $K_{4,(E)}$ based on $K_{2,(E)}$		
$SK_{S,(E)} \leftarrow K_{3,(E)}^{-1} K_{4,(E)} K_{1,(E)}^{-1}$		
$L_{(E)} \leftarrow K_{1,(E)} K_{2,(E)} K_{3,(E)}$	$\xrightarrow{L_{(E)}, E}$	Generates r and M_r
		$T \leftarrow SK_{S,(CA)} M_r$
		$B \leftarrow TL_{(E)}$
$B, E \leftarrow Decode(C_E)$	$\xleftarrow{r, C_E}$	$C_E \leftarrow Encode(B, E)$
$T \leftarrow BSK_{S,(E)}$		
$h \leftarrow H(H(C_E) \| H(C_{CA}))$		
Generates M_r, M_h from r, h		
$q \leftarrow \{M_h L_{(CA)} T K_{1,(E)}, K_{2,(E)}, K_{3,(E)}\}$		
$Q \leftarrow M_h L_{(CA)} B$		

Note: In this diagram, E represents identity information for EE.

Figure 1: Interaction between EE and CA in the PQCMC Process

1. **Key Generation**:
   - **EE**: Generates a set of matrices that constitute the secret key, the public key, as well as the signing secret key.
   - **CA**: Independently generates a random number r and a corresponding invertible matrix M_r, which are used in the certificate generation process.
2. **Certificate Generation**:
   - The CA computes T using the EE's public key and its own secret key component: $T = SK_{S,(E)} M_r$.
   - The reconstruction value B is then computed as $B = TL_{(E)}$.
   - The implicit certificate C_E is encoded from B and identity information E provided by the EE.

3. **Transmission and Verification**:
   - The CA sends C_E and the random number r back to the EE.
   - Other entities can use the certificate to verify EE's public key by first computing $h = H(H(C_E)\|H(C_{CA}))$, where C_{CA} is the CA's certificate. One can then reconstruct the EE's public key Q as $Q = M_h L_{(CA)} B$, where the invertible matrix M_h is based on h.
4. **Private and Public Key Updates**:
   - Upon receiving C_E and r, the EE calculates $h = H(H(C_E)\|H(C_{CA}))$.
   - The EE updates its secret key q as $q = \{M_h L_{(CA)} T K_{1,(E)}, K_{2,(E)}, K_{3,(E)}\}$ and the public key Q as $Q = M_h L_{(CA)} B$.

This method enables issuance of certificates without requiring the traditional CA signature, instead using cryptographic transformations to ensure authenticity and integrity.

3.3 Intended Security Properties

The PQCMC scheme is designed with several key security properties in mind, aiming to address both conventional and quantum cryptographic threats:

1. **Quantum Resistance:** PQCMC leverages the McEliece cryptography framework, renowned for its resistance to quantum attacks. Unlike cryptosystems based on factorization or discrete logarithms, which are vulnerable to quantum algorithms like Shor's algorithm, McEliece is based on the hard problem of decoding random linear codes, which currently offers no efficient quantum algorithm for its compromise.
2. **Integrity and Authenticity:** The integrity of certificates within the PQCMC framework is maintained through the cryptographic binding of certificate data with the CA's secret operations. Only the CA, with access to its private cryptographic components (like secret keys and random matrices), can generate valid certificates. Any unauthorized modification to a certificate disrupts the delicate cryptographic structure, making it impossible to reconstruct the correct public key using the CA's public components.
3. **Efficiency:** By utilizing implicit certificates, the PQCMC scheme significantly reduces the computational load associated with traditional signature verification processes. This approach is particularly advantageous in constrained environments, enhancing the system's scalability and responsiveness by minimizing the computational steps required for verifying certificate authenticity.
4. **Non-repudiation:** Despite the certificates being implicit, the PQCMC design ensures non-repudiation through cryptographic traces that irrevocably link the certificate to its issuer (the CA) and the recipient (the EE). Once a certificate is issued, neither the issuer can deny its issuance nor can the recipient deny its receipt. This is achieved through the unique cryptographic linkage between the issued certificate's data and the involved parties' keys.

These properties collectively aim to fortify the PQCMC scheme against emerging threats and position it as a viable solution for securing communications in the impending post-quantum era. The subsequent sections delve into the specific vulnerabilities discovered within this framework, presenting a critical examination of its robustness and the implications of these security flaws.

4 Security Analysis

The security analysis reveals a critical vulnerability within the PQCMC scheme: the inadvertent leakage of the CA's private signing key. This flaw allows an EE to derive and potentially misuse the CA's signing key, undermining the scheme's integrity and authenticity.

4.1 Technical Mechanism of the vulnerability

The vulnerability exploits the transmission of certain elements and their algebraic properties within the PQCMC protocol. Specifically:

1. The CA transmits a random number r to the EE, from which the invertible matrix M_r and its inverse M_r^{-1} are deterministically derived.

2. The CA then multiplies its private signing key by M_r to create T, which is sent to the EE as part of the certificate generation process.

3. Given T and M_r, the EE can algebraically manipulate these elements to retrieve $SK_{s,(CA)}$ by calculating $T \times M_r^{-1}$. This calculation effectively cancels out M_r, leaving the CA's private signing key exposed.

4.2 Conceptual Illustration of the Vulnerability

To better understand the vulnerability, consider the following conceptual representation:

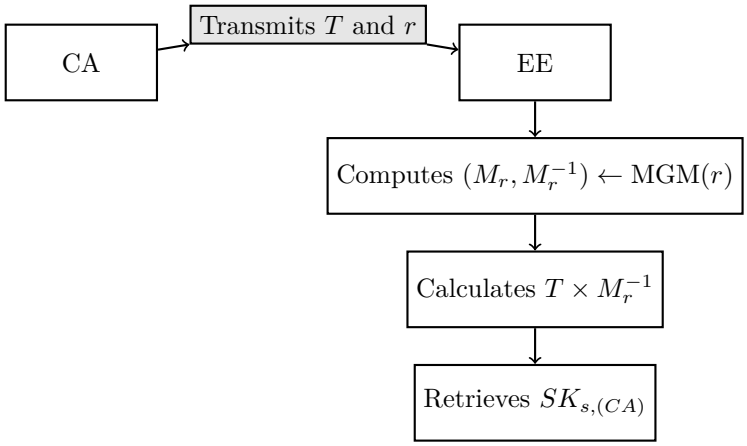


Figure 2: Diagrammatic representation of the vulnerability exploitation process in the PQCMC scheme.

As shown in Figure 2, the vulnerability stems from the EE's ability to:

1. Receives T and r from the CA.

2. Retrieves M_r along with M_r^{-1} using the received r through the function MGM$(\cdot)$ mentioned in Section 3.

3. Retrieves CA's private signing key by calculating $SK_{s,(CA)} = T \times M_r^{-1}$.

4.3 Impact Assessment

The ability of an EE to derive the CA's private signing key fundamentally compromises the cryptographic assurances of authenticity and integrity. This vulnerability permits unauthorized certificate forgery, thus potentially enabling impersonation or unauthorized access to secure systems. A summary of the security implications is presented in Table 2.

Table 2: Summary of Security Implications of the PQCMC Vulnerability

Impact	Description
Authenticity	Forged certificates cannot be distinguished from legitimate ones, enabling impersonation and unauthorized access.
Integrity	Integrity of the cryptographic system is compromised, allowing data manipulation and mistrust in transmitted information.
Non-repudiation	The ability to deny transactions facilitated by forged certificates could complicate legal and audit processes.

The implications of the PQCMC vulnerability extend beyond mere technical failures, impacting strategic and operational facets of entities employing this scheme. The urgency for corrective measures and a revised approach to cryptographic design and validation is evident and necessary to mitigate these risks.

5 Conclusion

The investigation into the PQCMC scheme has uncovered a significant vulnerability that undermines its security integrity. Our analysis revealed that an EE can potentially derive the CA's private signing key, compromising authenticity and integrity in the system. This vulnerability stems from PQCMC's use of invertible matrices and the transmission of the random number used to generate them. Importantly, the root cause is that PQCMC's signing mechanism is essentially a reversed McEliece encryption scheme. Such direct adaptation of encryption for signing is often problematic in cryptography, as demonstrated by this vulnerability in PQCMC.

This discovery is particularly disconcerting, given the increasing reliance on cryptographic solutions designed to withstand both classical and quantum computational threats. It underscores the critical need for rigorous, dynamic security evaluations in the development of post-quantum cryptographic protocols.

Moving forward, it is imperative that the cryptographic community continues to scrutinize and refine post-quantum schemes. This includes not only ensuring their resistance to quantum attacks but also verifying their security against sophisticated conventional exploits that could compromise their foundational principles.

While the PQCMC scheme shows promise in addressing post-quantum security needs, our findings emphasize the need for further refinement and validation of such schemes. The cryptographic community must remain vigilant, continuously evaluating and improving upon proposed solutions to ensure the security of our digital infrastructure in the face of advancing computational capabilities.

Acknowledgments

This research was supported by the National Science and Technology Council (NSTC), Taiwan (ROC), under Project Numbers NSTC 112-2221-E-004-004-, NSTC 113-2221-E-004-013- and NSTC 112-2634-F-004-001-MBK.

References

[1] Carlisle Adams and Steve Lloyd. *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Sams Publishing, 1999.

[2] Martin R Albrecht, Daniel J Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo Von Maurich, Rafael Misoczki, Ruben Niederhagen, et al. Classic mceliece: conservative code-based cryptography. 2022.

[3] Kooshiar Azimian, Javad Mohajeri, Mahmoud Salmasizadeh, and Samuel S Wagstaff Jr. Provable partial key escrow. *Int. J. Netw. Secur.*, 10(2):121–124, 2010.

[4] Mihir Bellare and Shafi Goldwasser. Verifiable partial key escrow. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 78–91, 1997.

[5] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In *IMA international conference on cryptography and coding*, pages 30–45. Springer, 1997.

[6] Daniel RL Brown, Robert Gallant, and Scott A Vanstone. Provably secure implicit certificate schemes. In *Financial Cryptography: 5th International Conference, FC 2001 Grand Cayman, British West Indies, February 19–22, 2001 Proceedings 5*, pages 156–165. Springer, 2002.

[7] Jehoshua Bruck and Moni Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, 1990.

[8] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. Openpgp message format. RFC 2440, RFC Editor, 11 1998.

[9] Matthew Campagna. SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV). *Standards for Efficient Cryptography*, 2013.

[10] Juan Caubet, Oscar Esparza, Juanjo Alins, Jorge Mata-Díaz, and Miguel Soriano. Securing identity assignment using implicit certificates in p2p overlays. In *Trust Management VII: 7th IFIP WG 11.11 International Conference, IFIPTM 2013, Malaga, Spain, June 3-7, 2013. Proceedings 7*, pages 151–165. Springer, 2013.

[11] Abel C H Chen. PQCMC: Post-Quantum Cryptography McEliece-Chen Implicit Certificate Scheme. *Cryptology ePrint Archive*, 2023.

[12] Santosh Chokhani, Wes Ford, and Tim Polk. Internet x.509 public key infrastructure certificate policy and certification practices framework. RFC 2527, RFC Editor, 3 1999.

[13] Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a mceliece-based digital signature scheme. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 157–174, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[14] Daniela Engelbert, Raphael Overbeck, and Arthur Schmidt. A summary of mceliece-type cryptosystems and their security. *Journal of Mathematical Cryptology*, 1(2):151–199, 2007.

[15] Jozef Gruska et al. *Quantum computing*, volume 2005. McGraw-Hill London, 1999.

[16] Mohammad Kamrul Hasan, Zhou Weichen, Nurhizam Safie, Fatima Rayan Awad Ahmed, and Taher M Ghazal. A survey on key agreement and authentication protocol for internet of things application. *IEEE Access*, 2024.

[17] Mark Horowitz and Emily Grumbling. Quantum computing: progress and prospects. 2019.

[18] Ray Hunt. Pki and digital certification infrastructure. In *Proceedings. Ninth IEEE International Conference on Networks, ICON 2001.*, pages 234–239. IEEE, 2001.

[19] Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, pages 107–124, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[20] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[21] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization, 2024. Accessed: 2024-06-13.

[22] Kim Thuat Nguyen, Maryline Laurent, and Nouha Oualha. Survey on secure communication protocols for the internet of things. *Ad Hoc Networks*, 32:17–31, 2015.

[23] Klaus Schmeh. *Cryptography and public key infrastructure on the Internet*. John Wiley & Sons, 2006.

[24] Michael Schukat and Pablo Cortijo. Public key infrastructures and digital certificates for the internet of things. In *2015 26th Irish signals and systems conference (ISSC)*, pages 1–5. IEEE, 2015.

[25] Savio Sciancalepore, Angelo Capossele, Giuseppe Piro, Gennaro Boggia, and Giuseppe Bianchi. Key management protocol with implicit certificates for iot systems. In *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pages 37–42, 2015.

[26] Mihai Serb and Mihai Togan. A certificate–based signature scheme for secure mobile communications. In *2010 8th International Conference on Communications*, pages 469–472. IEEE, 2010.

[27] P Shor. Proceedings of the 35th annual symposium on the foundations of computer science, ieee. 1994.

[28] Peter W Shor. Quantum computing. *Documenta Mathematica*, 1(1000):467–486, 1998.

[29] Sujan Raj Shrestha and Young-Sik Kim. New mceliece cryptosystem based on polar codes as a candidate for post-quantum cryptography. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pages 368–372. IEEE, 2014.

[30] Valmiki Siddhartha, Gurjot Singh Gaba, and Lavish Kansal. A lightweight authentication protocol using implicit certificates for securing iot systems. *Procedia Computer Science*, 167:85–96, 2020.

[31] Ahcene Teniou and Boucif A Bensaber. Efficient and dynamic elliptic curve quvanstone implicit certificates distribution scheme for vehicular cloud networks. *Security and Privacy*, 1(1):e11, 2018.

[32] Don J Torrieri. *Principles of secure communication systems*. Artech House, INC., 1992.

[33] CH Ugwuishiwu, UE Orji, CI Ugwu, and CN Asogwa. An overview of quantum cryptography and shor's algorithm. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(5), 2020.

[34] Joel Weise. Public key infrastructure overview. *Sun BluePrints OnLine, August*, pages 1–27, 2001.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 75–96.

A Cryptanalysis on the Bivariate Cryptosystem in a Multivariate Setting

Siti Nabilah Yusof[1], Muhammad Rezal Kamel Ariffin[1,2] and Wan Nur Aqlili Ruzai[3]

[1] Institute for Mathematical Research, Universiti Putra Malaysia, 43400 Serdang Selangor, Malaysia. sitinabilahyusof@gmail.com

[2] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang Selangor, Malaysia. rezal@upm.edu.my

[3] School of Distance Education, Universiti Sains Malaysia, 11800 Penang, Malaysia. aqliliruzai@usm.my

Abstract. In 1999, the Polynomial Reconstruction Problem (PRP) was introduced as a novel hard problem in post–quantum cryptography. The first cryptographic system based on a univariate PRP was designed by Augot and Finiasz; it was presented at Eurocrypt 2003 and eventually broken in 2004. A bivariate PRP was suggested in 2013. The layout has been altered from the original concept of Augot and Finiasz. In this paper, we present an extended version of the bivariate PRP cryptosystem in a multivariate setting and utilize the Coron methods and the modified Berlekamp–Welch algorithm. Our strategic approach enabled us to obtain the secret parameter of the extended version of the bivariate PRP cryptosystem. We concluded that the extended version of the bivariate PRP cryptosystem is not secure against the Indistinguishable Chosen–Plaintext Attack (IND–CPA) as a result of this finding.

Keywords: Polynomial Reconstruction Problem · Univariate Polynomial · Bivariate Polynomial · Multivariate Polynomial · Indistinguishable Chosen–Plaintext Attack

1 Introduction

A cryptosystem must include a hard mathematical problem to be effective and secure. Shor created an algorithm in 1994 that makes the popular number theoretical asymmetric cryptosystem used today unsecured when run on a full-fledged quantum computer [23]. These systems are based on the Integer Factorization Problem and the Discrete Logarithm Problem. El Gamal, RSA, Diffie-Hellman, and Elliptic Curve Cryptography are a few of the well-known algorithms [13]. In addition, a search for quantum-resistant algorithms has been requested by the National Institute of Standards and Technology (NIST) [6].

Post-quantum cryptography is desirable these days for information security. A cryptographic algorithm known as post-quantum cryptography is thought to resist attacks from quantum computers [12]. Furthermore, post-quantum cryptography can be divided into five primary categories: hash-based, multivariate, lattice, code and isogeny [10]. The primary purpose of post-quantum cryptography is to establish measures to prevent attacks from quantum computers [8]. Cryptosystem designers must consider the time complexity and memory requirements for an attack to guarantee the security of their systems [19]. According to [17], these characteristics are essential for verifying the security of a cryptographic scheme. The website Quantum Algorithm Zoo lists many mathematical problems with a probability of withstanding an attack by a quantum computer. Polynomial Reconstruction Problem (PRP) is one of the problems listed in Quantum Algorithm Zoo.

This problem was first presented as a hard new problem for cryptography design in 1999 [4].

In decoding formulation, this PRP is equivalent to Reed-Solomon error-correcting codes [20, 21]. Evidence that PRP is resistant to quantum computers makes it an excellent option for post-quantum cryptography [15]. Moreover, the PRP offers benefits in terms of efficacy and efficiency.

Consider that the error weight, w, satisfies the equation $w \leq \frac{n-k}{2}$, in which k represents the degree of a polynomial and n represents the number of elements in a vector. In that situation, PRP is solvable in polynomial time. This has been improved from $w \leq \frac{n-k}{2}$ to $w \leq n - \sqrt{kn}$ [9]. In 2003, Augot and Finiasz [3] developed a PRP-based cryptosystem that we refer to as AF–Cryptosystem. This cryptosystem uses a univariate polynomial [14, 16]. The AF–Cryptosystem used two types of PRP; the first is explained in the [11], and the second was created especially to ensure that the decryption process would be successful. We refer to the second PRP as Augot and Finiasz Solvable PRP (AF–SPRP) and define it as follows:

Definition 1. ***(Augot and Finiasz Solvable PRP)*** Given n, k, t and $(x_i, y_i)_{i=1,\cdots,n}$, output any polynomial p such that $deg < k$ and $p(x_i) = y_i$ for at least t values of i where $t = n - w$.

According to Definition 1, the decryption procedure in AF–Cryptosystem can occur. One has to output a polynomial that satisfies every point on a Cartesian plane after receiving t points. The number of elements in a vector that are zero is indicated by the value of the parameter t. The decryption process is completed using Lagrange interpolation.

Nevertheless, Coron [7] was able to decrypt the AF–Cryptosystem completely. Later, in 2013, Ajeena et al. developed a modified version of AF–Cryptosystem, utilizing the Vandermonde matrix and bivariate polynomials in their design. This improved cryptosystem is now known as the AAK–Cryptosystem. The AAK–Cryptosystem creators believed that increasing the number of variables would enhance the system's security and make it more resistant to attacks.

Cryptosystem designers typically claim security in terms of the exponential time and memory required for attacks [17]. It is a necessary feature to confirm the security of a cryptographic scheme [19]. However, it should be noted that indistinguishability is also a necessary feature of a cryptosystem that might be selected to be applied to a non-exponentially large plaintext domain. A design must be protected against the Indistinguishable Chosen-Plaintext Attack (IND–CPA) secure in order to defeat an attacker who may re-encrypt every conceivable plaintext and compare it to the ciphertext.

If every Probabilistic Polynomial Time Adversary (PPTA) has a little "advantage" over random guessing, then the cryptosystem is IND-CPA safe. An adversary cannot win the IND-CPA game with a probability greater than $\frac{1}{2} + \varepsilon(n)$ in an IND-CPA-secure cryptosystem, where $\varepsilon(n)$ is a negligible function in security parameter n. This research on the extended version of AAK-Cryptosystem in a multivariate setting is intended to determine whether or not it is IND-CPA secure.

Our contribution. This study is an expansion of our previously published work in [25, 24], in which we demonstrate that the AAK-Cryptosystem is not safe against the IND–CPA and provide details for recovering the secret key within the system. This paper presents an extended version of the AAK-Cryptosystem in a multivariate setting. Our incentive originates from Coron's cryptanalysis of the univariate PRP-based cryptosystem. We employ the same technique as the AAK-Cryptosystem and show that the extended version of the AAK-Cryptosystem in a multivariate setting is not IND-CPA secure.

Organization of the article. The following is an outline of this paper. The foundations of PRP, IND-CPA, Vandermonde matrices, AAK–Cryptosystem, multivariate polynomial and the extended version of the AAK–Cryptosystem in multivariate settings

are presented in Section 2. We provide a secure implementation of AAK–Cryptosystem in a multivariate setting in Section 3. We summarize our work in Section 4.

2 Materials and Methods

This section provides the fundamentals of PRP, Vandermonde matrices, AAK–Cryptosystem, multivariate polynomial and the extended version of the AAK–Cryptosystem in multivariate settings that will be used in the rest of the paper.

2.1 Polynomial Reconstruction Problem (PRP)

In this section, we review the fundamental concepts of PRP. It is worth mentioning that the generalized Reed–Solomon list decoding problem is simplified to it [22]. We then present the PRP definition based on [11].

Definition 2. ***(PRP from [11])*** Let $p(x) = a_k x^k + \cdots + a_1 x + a_0$ be a secret polynomial over finite field $\mathbb{F}_q$. One is given access to the oracle and query values of $x_i \in \mathbb{F}_q$ where $1 \leq i \leq k+1$. Then one needs to output the coefficients $a_0, a_1, \ldots, a_k$ to obtain $p(x)$.

Definition 2 makes clear that an oracle will output $p(y)$ when it receives a value of $y \in \mathbb{F}_q$. Finding the coefficients $a_0, a_1, \ldots, a_k$ is the primary objective in solving PRP [11]. Traditionally, to find the number of coefficients, $k+1$ queries are required. The PRP has a query complexity of $\mathcal{O}\binom{k+1}{k}$ for univariate polynomials of degree k.

2.1.1 PRP Computational Complexity

The highest possible degree of polynomial $p(x)$ is k. Since there are a total of $k+1 = q-1$ coefficients, this results in $k = q-2$. Hence,

$$\mathcal{O}\binom{k+1}{k} = \mathcal{O}(q-1).$$

Finding the value of x is not feasible if $q \approx 2^n$. As a result, solving PRP requires $\mathcal{O}\binom{k+1}{k}$.

2.2 Indistinguishable Chosen-Plaintext Attack (IND–CPA)

To prevent attacks on the cryptographic protocol, every cryptosystem must have its fundamental security requirements examined, particularly its indistinguishability features [26]. A security concept known as Indistinguishable under Chosen-Plaintext Attack (IND-CPA) applies to cryptosystems in which a random oracle is contacted by a PPTA during a two-phase session, known as the learning and challenge phases [5]. The following defines IND-CPA:

Definition 3. ***(Indistinguishable under Chosen-Plaintext Attack)*** The IND-CPA security model is defined by the following game between random oracle and PPTA:

1. The random oracle creates a cryptographic system, produces $(PK, SK) = Gen(1^n)$, selects a random $b \in \{0, 1\}$, and publishes the public key PK while keeping the secret key SK private.

2. The PPTA selects two messages (μ_0 and μ_1) and sends them to the random oracle.

3. The random oracle chooses one of the two messages at random and encrypts it before sending the ciphertext $C = enc(\mu_b, PK)$ to the PPTA.

4. The PPTA determines b'. If $b' = b$, then it outputs 1; else, 0.

If there is a negligible function $\varepsilon(n)$ for every PPTA, then a cryptosystem is indistinguishable under the Chosen-Plaintext Attack, where

$$Pr(b' = b) \leq \frac{1}{2} + \varepsilon(n).$$

An IND-CPA-secure cryptosystem prevents passive adversaries from obtaining encrypted message information during communication between two parties [1].

2.3 Vandermonde Method

This method can find an interpolating polynomial in two or more dimensions using a Vandermonde matrix, also known as an alternant matrix. Consider the subsequent collection of two-dimensional points: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$; let $z_1, z_2, \ldots, z_n$ be the points from which values must be obtained. Finding a bivariate polynomial of degree $n-1$ that fits every data point is required. The steps for this method are outlined as follows:

1. Write the general formula for the bivariate polynomial of degree $n-1$.

2. Determine the polynomial at the following points: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

3. Compute the solution of linear equation system.

The problem can be expressed simply as $V \cdot c = Z$, where z values are vectors Z and c is the vector of coefficients. The AAK-Cryptosystem decoding process employs this method.

2.4 AAK-Cryptosystem

The bivariate PRP–based cryptosystem proposed by Ajeena et al. is explained below [2]. The AAK–Cryptosystem considered the value of the parameter n, which denotes the number of elements in a vector, along with the values for the following parameters:

Table 1: Parameters used in AAK–Cryptosystem.

Parameter	Remark
$\mathbb{F}_q$	A finite field of size q
n	The quantity of elements within a vector
k	Its dimension
W	The big error vector, E's weight when PRP is difficult which, $W > \frac{n-k}{2}$ [2]
w	The small error, e's weight which gives PRP the ability to decrypt the ciphertext where $w \leq \frac{n-k}{2}$ [7]

Remark 1. The highest possible number of nonzero items in a vector is represented by the value w.

Remark 2. The value $n - w$ is the number of zeros in a vector.

The designers in [2] created a cryptosystem with the following algorithms by using the above listed parameters:

Algorithm 1 Key Generation Process

Input: Parameters $(q,\ n,\ k,\ W,\ w)$
Output: Public Key, PK and SK pair (C, E)

1: Alice covertly creates a monic bivariate polynomial $p(X, Y)$ with a degree of $k-1$ for X and Y, an error vector E with the weight of W.
2: After computing the codeword $C = ev(p(X, Y)) = p(x_i, y_i)$, where $x_i, y_i \in \mathbb{F}_q$, Alice calculates $PK = C + E$.
3: The public key, PK and secret key pair (C, E) are output.

Algorithm 2 Encryption Process

Input: Message, $\mu \in \mathbb{F}_q$
Output: Ciphertext, CT

1: Bob wishes to transmit a $k+1$-length polynomial message $\mu(X, Y)$.
2: By computing $\mu = ev(\mu(X, Y)) = \mu(x_i, y_i)$, the message is encoded into a codeword μ.
3: Bob constructs a small error vector e with a weight of w and $\alpha \in \mathbb{F}_q$ at random.
4: Alice receives ciphertext $CT = \mu + \alpha \times PK + e$ from Bob.

2.4.1 Proof of Correctness

In this section, we present a proof of correctness for the decryption algorithm in AAK-Cryptosystem since the proof was not presented in [2].

Proposition 1. *The message polynomial $\mu(X, Y)$ can be retrieved via the AAK-Cryptosystem decryption algorithm.*

Proof. We proof that when we have the ciphertext CT, we can retrieve the message $\mu(x, y)$, observe that:

$$\begin{aligned} CT &= \mu + \alpha \times PK + e \\ &= \mu + \alpha \times (C + E) + e. \end{aligned} \tag{1}$$

From E, we find the position of zero elements. Consider the shortened codes for e, C, μ and CT are $\overline{e}$, $\overline{C}$, $\overline{\mu}$ and $\overline{CT}$ respectively. Then, equation (1) turns out to be

$$\overline{CT} = \overline{\mu} + \alpha \times \overline{C} + \overline{e}. \tag{2}$$

Based on (2), $\overline{\mu} + \alpha \times \overline{C} \in \overline{RS_k}$. Correct $\overline{CT}$ to obtain $\tilde{\mu} + \alpha \times \tilde{C}$, since the weight of e is less than the error correction capacity $\overline{RS_k}$. Using the Vandermonde method, a unique polynomial $q(x, y)$ to the power of $k-1$ is computed; as a result, we obtain

$$ev(q(x_i, y_i)) = \tilde{\mu_i} + \alpha \times \tilde{C}_i \tag{3}$$

for $i \in \{1, 2, \ldots, n\}$. We know that $ev(q(x_i, y_i)) = q(x_i, y_i)$, thus, $\tilde{C}$ is equivalent to $ev(p(x_i, y_i)) = p(x_i, y_i)$ and $\tilde{\mu}$ is equivalent to $ev(\mu(x_i, y_i)) = \mu(x_i, y_i)$, therefore

$$\begin{aligned} q(x_i, y_i) &= \mu(x_i, y_i) + \alpha p(x_i, y_i) \\ \mu(x_i, y_i) &= q(x_i, y_i) - \alpha p(x_i, y_i). \end{aligned} \tag{4}$$

From here, we can retrieve message polynomial $\mu(x, y)$. □

Algorithm 3 Decryption Process

Input: Ciphertext, CT

Output: Message polynomial, $\mu(X, Y)$

1: Alice determines $\overline{CT} = \overline{\mu} + \alpha \times \overline{C} + \overline{e}$ for i where $E_i = 0$.
2: The $\tilde{CT} = \tilde{\mu} + \alpha \times \tilde{C}$ can be retrieved where Alice corrects the $\overline{CT}$.
3: Alice evaluates a unique polynomial $q(X, Y)$ with the degree of $k - 1$ using the Vandermonde method.
4: Alice identifies the initial coefficient with the highest degree in $q(X, Y)$ where the initial coefficient is the value of α.
5: Polynomial $\mu(X, Y)$ can be retrieved when Alice computes $q(X, Y) - \alpha p(X, Y)$.

2.5 Multivariate Polynomial

According to [18], a function defined by a polynomial is a polynomial function. Let f be a function from real numbers $\mathbb{R}$ to $\mathbb{R}$, defined as $f(x) = 8x^2 + 2x + 5$, a one-variable polynomial. As in the case of $f(x, y, z) = 3yz + 2x + 4y + z + 7$, a polynomial with multiple variables is a polynomial function having multiple inputs. A multivariate polynomial, also known as a multilinear polynomial, is linear in each variable independently but not simultaneously. The following is a definition of a multivariate polynomial:

Definition 4. A multivariate polynomial of degree $k - 1$ with its corresponding constant coefficients, δ's is given by

$$\begin{aligned} f(\chi) &= f(x_1, x_2, x_3, \ldots, x_\beta) \\ &= \delta_{i_1 i_2 i_3 \ldots i_{k-1} i_k} x_1^{i_1} x_2^{i_2} x_3^{i_3} \ldots x_{\beta-1}^{i_{k-1}} x_\beta^{i_k} + \ldots + \delta_{111\ldots11} x_1^1 x_2^1 x_3^1 \ldots x_{\beta-1}^1 x_\beta^1 \\ &\quad + \delta_{111\ldots10} x_1^1 x_2^1 x_3^1 \ldots x_{\beta-1}^1 x_\beta^0 + \delta_{111\ldots01} x_1^1 x_2^1 x_3^1 \ldots x_{\beta-1}^0 x_\beta^1 \\ &\quad + \delta_{1000\ldots00} x_1^1 x_2^0 x_3^0 \ldots x_{\beta-1}^0 x_\beta^0 + \delta_{0111\ldots11} x_1^0 x_2^1 x_3^1 \ldots x_{\beta-1}^1 x_\beta^1 \\ &\quad + \delta_{0111\ldots10} x_1^0 x_2^1 x_3^1 \ldots x_{\beta-1}^1 x_\beta^0 + \ldots + \delta_{000\ldots01} x_1^0 x_2^0 x_3^0 \ldots x_{\beta-1}^0 x_\beta^1 + \delta_{000\ldots0} \end{aligned}$$

where i_κ represents the exponent for each variable in the particular monomial where $i_\kappa \in \{0, 1, \ldots, k-1\}$. Next, β is the number of variables in a polynomial and χ represents the variable vector of a polynomial.

The multivariate polynomial used in this research is a monic multivariate polynomial and has the highest degree up to $k - 1$ with respect to all variables in χ.

Example 1. Suppose $\beta = 2$ and $k = 2$. From Definition 4, we will produce a 2 variable polynomial of degree 1. That is,

$$f(x, y) = \delta_{11} xy + \delta_{10} x + \delta_{01} y + \delta_{00}$$

where δ is the constant coefficient and $i_\kappa \in \{0, 1\}$.

Example 2. Suppose $\beta = 3$ and $k = 2$. From Definition 4, we will produce a 3 variable polynomial of degree 1. That is,

$$f(x, y, z) = \delta_{111} xyz + \delta_{110} xy + \delta_{101} xz + \delta_{100} x + \delta_{011} yz + \delta_{010} y + \delta_{001} z + \delta_{000}$$

where δ is the constant coefficient and $i_\kappa \in \{0, 1\}$.

Example 3. Suppose $\beta = 3$ and $k = 3$. From Definition 4, we will produce a 3 variable polynomial of degree 2. That is,

$$\begin{aligned} f(x, y, z) = {} & \delta_{222} x^2 y^2 z^2 + \delta_{221} x^2 y^2 z + \delta_{220} x^2 y^2 + \delta_{212} x^2 y z^2 + \delta_{211} x^2 yz + \delta_{202} x^2 z^2 \\ & + \delta_{122} x y^2 z^2 + \delta_{121} x y^2 z + \delta_{112} xyz^2 + \delta_{111} xyz + \delta_{110} xy + \delta_{101} xz \\ & + \delta_{100} x + \delta_{022} y^2 z^2 + \delta_{011} yz + \delta_{010} y + \delta_{001} z + \delta_{000} \end{aligned}$$

where δ is the constant coefficient and $i_\kappa \in \{0, 1, 2\}$.

2.6 The Extended Version of the AAK-Cryptosystem in Multivariate Setting

In this section, we described the algorithm for multivariate PRP cryptosystem based on the AAK-Cryptosystem. The following is the algorithm:

Algorithm 4 Key Generation Process

Input: Parameters (q, n, k, W, w)
Output: Public Key, PK and SK pair (C, E)

1. Alice secretly generates monic multivariate polynomial $p(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ of degree equals to $k-1$ with respect to all variables and big error vector E with the weight, W.
2. Alice calculates codeword $C = ev(p(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)) = p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ where $x_{1,i}, x_{2,i}, \ldots, x_{\beta,i} \in \mathbb{F}_q$ and calculates $PK = C + E$.
3. Publish public key, PK secret key pair (C, E).

Algorithm 5 Encryption Process

Input: Message, $\mu \in \mathbb{F}_q$
Output: Ciphertext, CT

1. Bob wants to deliver a message polynomial $\mu(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ with length $k+1$.
2. The message is encoded into a codeword μ by calculating $\mu = ev(\mu(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)) = \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$.
3. Bob randomly generates $\alpha \in \mathbb{F}_q$ and small error vector e with the weight, w.
4. Bob calculates ciphertext $CT = \mu + \alpha \times PK + e$ and delivers the ciphertext to Alice.

Algorithm 6 Decryption Process

Input: Ciphertext, CT
Output: Message, $\mu \in \mathbb{F}_q$

1. For i, where $E_i = 0$, determine $\overline{CT} = \overline{\mu} + \alpha \times \overline{C} + \overline{e}$.
2. Correct $\overline{CT}$ to obtain $\tilde{CT} = \tilde{\mu} + \alpha \times \tilde{C}$.
3. Compute unique polynomial $q(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ of degree $k-1$ by using Vandermonde method.
4. Determine the leading coefficient $q(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$.
5. Compute $\mu(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta) = q(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta) - \alpha p(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$.

2.6.1 Proof of Correctness for the AAK-Cryptosystem in Multivariate Setting

We presented the proof of correctness for multivariate PRP cryptosystem based on the AAK-Cryptosystem. The following is the proof of correctness:

Proposition 2. *The decryption algorithm for the AAK-Cryptosystem in a multivariate setting is correct.*

Proof. We want to prove that from the ciphertext CT, the message $\mu(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ can be obtained, observe the following:

$$\begin{aligned} CT &= \mu + \alpha \times PK + e \\ &= \mu + \alpha \times (C + E) + e. \end{aligned} \tag{5}$$

Observe the position $E_i = 0$. Given that $\overline{\mu}$, $\overline{C}$, $\overline{e}$ and $\overline{CT}$ represent as the shortened code for μ, C, e and CT respectively. Now, (5) becomes

$$\overline{CT} = \overline{\mu} + \alpha \times \overline{C} + \overline{e}. \tag{6}$$

From (6), $\overline{\mu} + \alpha \times \overline{C} \in \overline{RS_k}$. Provided that e has the weight that is less than error correction capacity $\overline{RS_k}$ then $\overline{CT}$ can be corrected and find $\tilde{\mu} + \alpha \times \tilde{C}$. The Vandermonde method is used to compute the unique polynomial $q(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ of degree $k-1$ and

$$ev(q(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})) = \tilde{\mu_i} + \alpha \times \tilde{C}_i \tag{7}$$

for $i \in \{1, 2, \ldots, n\}$. Since we know that $ev(q(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})) = q(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$, $\tilde{C} = ev(p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})) = p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and $\tilde{\mu} = ev(\mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})) = \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ then

$$\begin{aligned} q(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) &= \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) + \alpha p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \\ \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) &= q(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) - \alpha p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}). \end{aligned} \tag{8}$$

From (8), the message $\mu(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X}_\beta)$ can be retrieved. □

3 Result

In this section, we present that the extended version of AAK–Cryptosystem is not IND–CPA secure. We also provide a numerical illustration.

3.1 Cryptanalysis of the Extended Version of AAK-Cryptosystem

Proposition 3. *Let $M(\lambda)$ be a matrix, $M'(\lambda)$ be the corresponding sub–square matrix of $M(\lambda)$ and polynomial $f(\lambda) = \det M'(\lambda)$. If the adversary can correctly ascertain value $\mu + e$, then given public key PK and ciphertext CT, the adversary can recover secret key, SK α in polynomial time.*

Proof. Let CT_i, PK_i and e_i be vector elements in CT, PK and e, respectively. That is,

$$CT_i = \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) + \alpha \cdot PK_i + e_i, \quad \forall\ 1 \leq i \leq n$$

and

$$PK_i = C_i + E_i, \quad \forall\ 1 \leq i \leq n.$$

Since the vector C is from the evaluation of polynomial $p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$. Note that from Algorithm 4, a monic polynomial $p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ has the highest power up to $k-1$ with respect to all the variables and the length of polynomial $\mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ must be $k+1$. Consider the following set of equations:

$$\exists\ V, \mu, \alpha \begin{cases} \deg(V) \leq k-1, \quad V \neq 0 \\ \forall i,\ V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot (CT_i - \alpha \times PK_i) \\ \qquad = V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot \mu(x_i, y_i, \ldots, \theta_i) \end{cases} \tag{9}$$

$$\exists\, V, N, \lambda \begin{cases} \deg(V) \leq k-1, \quad V \neq 0, \quad \deg(N) \leq k-1 \\ \forall i,\ V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot (CT_i - \lambda \times PK_i) \\ \qquad = N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \end{cases} \tag{10}$$

it is clear within (10) that when $\lambda = \alpha$ one will have $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) = \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$. For a given λ, equation (10) gives $2^{\beta-1}k^2$ unknowns that are the coefficients of polynomials $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and β is the number of variables in a polynomial where $\beta \neq 1$. Hence, $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ are as follows,

$$\begin{aligned} V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) = {} & v_{2^{\beta-2}k^2-1} x_{1,i}^{k-1} x_{2,i}^{k-1} \ldots x_{\beta,i}^{k-1} + \ldots + v_3 x_{1,i}^1 x_{2,i}^0 \ldots x_{\beta,i}^0 \\ & + v_2 x_{1,i}^0 x_{2,i}^1 \ldots x_{\beta,i}^0 + v_1 x_{1,i}^0 x_{2,i}^0 \ldots x_{\beta,i}^1 + v_0 x_{1,i}^0 x_{2,i}^0 \ldots x_{\beta,i}^0 \end{aligned}$$

$$\begin{aligned} N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) = {} & n_{2^{\beta-2}k^2-1} x_{1,i}^{k-1} x_{2,i}^{k-1} \ldots x_{\beta,i}^{k-1} + \ldots + n_3 x_{1,i}^1 x_{2,i}^0 \ldots x_{\beta,i}^0 \\ & + n_2 x_{1,i}^0 x_{2,i}^1 \ldots x_{\beta,i}^0 + n_1 x_{1,i}^0 x_{2,i}^0 \ldots x_{\beta,i}^1 + n_0 x_{1,i}^0 x_{2,i}^0 \ldots x_{\beta,i}^0 \end{aligned}$$

which the coefficients for $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ can be represented as the coordinate vector, Y where

$$Y^T = (v_0, v_1, \ldots, v_{2^{\beta-2}k^2-1}, n_0, n_1, \ldots, n_{2^{\beta-2}k^2-1}). \tag{11}$$

Next, a matrix $M(\lambda)$ is created by the following entries,

$$M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = (CT_i - \lambda \cdot PK_i) \cdot x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \tag{12}$$

and

$$M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = -x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \tag{13}$$

where $i \in \{1, \ldots, n\},\ a_1 \in \{0, \ldots, k-1\},\ a_2 \in \{0, \ldots, k-1\}, \ldots, a_\beta \in \{0, \ldots, k-1\}$ for (12) and (13). Equation (12) is used for the first half columns of $M(\lambda)$ where the values of a and b are based on consecutive exponents of each monomial from polynomial $p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$. Next, equation (13) is used for the next half columns of $M(\lambda)$ where the values of a and b are based on consecutive of each monomial from polynomial $p(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ as well. Hence, $M(\lambda)$ can be a rectangular or a square matrix.

Let $M(\lambda)$ be a matrix with dimension $r \times s$. There are two cases for a rectangular matrix where

i) For the case of $r > s$, select a $s \times s$ sub-square matrix $M'(\lambda)$ in $M(\lambda)$.

ii) For the case of $r < s$, select a $r \times r$ sub-square matrix $M'(\lambda)$ in $M(\lambda)$.

By using equations (12) and (13), and with numerical input of public values $(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$, we construct $M(\lambda)$, where λ represents the possible value of α. From equation (12), we have

$$\begin{aligned} M(\lambda)_{1,0,0,\ldots,0} &= (CT_1 - \lambda \cdot PK_1) \cdot (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1}^0 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^0 \\ &= (CT_1 - \lambda \cdot PK_1) \end{aligned}$$

$$\begin{aligned} M(\lambda)_{1,1,0,\ldots,0} &= (CT_1 - \lambda \cdot PK_1) \cdot (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1}^1 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^0 \\ &= (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1} \end{aligned}$$

$$\begin{aligned} M(\lambda)_{1,0,1,\ldots,0} &= (CT_1 - \lambda \cdot PK_1) \cdot (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1}^0 \cdot x_{2,1}^1 \cdot \ldots \cdot x_{\beta,1}^0 \\ &= (CT_1 - \lambda \cdot PK_1) \cdot x_{2,1} \end{aligned}$$

$$\vdots$$

$$\begin{aligned} M(\lambda)_{1,0,0,\ldots,1} &= (CT_1 - \lambda \cdot PK_1) \cdot (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1}^0 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^1 \\ &= (CT_1 - \lambda \cdot PK_1) \cdot x_{\beta,1} \end{aligned}$$

$$\vdots$$

$$M(\lambda)_{n,k-1,k-1,k-1,\ldots k-1} = (CT_n - \lambda \cdot PK_n) \cdot x_{1,n}^{k-1} \cdot x_{2,n}^{k-1} \cdot \ldots \cdot x_{\beta,n}^{k-1}$$

From equation (13), we have

$$M(\lambda)_{1,0,0,\ldots,0} = -x_{1,1}^0 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^0 = -1 \pmod q$$

$$M(\lambda)_{1,1,0,\ldots,0} = -x_{1,1}^1 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^0 = -x_1 \pmod q$$

$$M(\lambda)_{1,0,1,\ldots,0} = -x_{1,1}^0 \cdot x_{2,1}^1 \cdot \ldots \cdot x_{\beta,1}^0 = -x_{2,1} \pmod q$$

$$\vdots$$

$$M(\lambda)_{1,0,0,\ldots,1} = -x_{1,1}^0 \cdot x_{2,1}^0 \cdot \ldots \cdot x_{\beta,1}^1 = -x_{\beta,1} \pmod q$$

$$\vdots$$

$$M(\lambda)_{n,k-1,k-1,k-1,\ldots k-1} = -x_{1,n}^{k-1} \cdot x_{2,n}^{k-1} \cdot \ldots \cdot x_{\beta,n}^{k-1} \pmod q$$

Next, we construct the matrix $M(\lambda)$ utilizing the above equations and we get $M(\lambda)$ in Appendix A. When equations (12) and (13) are multiplied by $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$, respectively, we have

$$\begin{aligned} V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = (CT_i - \lambda \cdot PK_i) \cdot x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot \\ x_{\beta,i}^{a_\beta} \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \end{aligned} \tag{14}$$

and

$$\begin{aligned} N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = -x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot \\ N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}). \end{aligned} \tag{15}$$

The summation of (14) and (15) is

$$\begin{aligned} &(CT_i - \lambda \cdot PK_i) \cdot x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) - \\ &x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}). \end{aligned} \tag{16}$$

Since $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) = \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$, (16) becomes

$$\begin{aligned} &(CT_i - \lambda \cdot PK_i) \cdot x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) - x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot \\ &\mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}). \end{aligned} \tag{17}$$

Equations (9) and (10) state that $\lambda = \alpha$ and $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot (CT_i - \alpha \times PK_i) = V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$; hence,

$$x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) - x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta} \cdot \mu(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) \cdot V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i}) = 0. \tag{18}$$

As a result of equation (12) multiplied by $V(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$ and equation (13) multiplied by $N(x_{1,i}, x_{2,i}, \ldots, x_{\beta,i})$, the summation of (14) and (15) is equal to 0 as stated in (14). By utilizing the matrix representation given by $M(\lambda)$, and the vector Y as defined in (11), equation (18) can be put forward as the following:

$$M(\lambda) \cdot Y = 0, \quad Y \neq 0. \tag{19}$$

There are two types of matrix $M(\lambda)$ which are rectangular matrix and square matrix. For rectangular matrix $M(\lambda)$ we need to follow cases (i) and (ii) to find sub-square matrix $M'(\lambda)$. Meanwhile, for square matrix $M(\lambda)$ we take $M(\lambda)$ as $M'(\lambda)$ to compute $f(\lambda) = \det(M'(\lambda))$. Next, a sub-square matrix $M'(\lambda)$ can be invertible when the determinant is not equal to 0. Next, we need to identify the parameter λ from the matrix $M'(\lambda)$, which was constructed by relations (9) and (10). Given relation,

$$M'(\lambda) \cdot Y = 0 \pmod{q} \tag{20}$$

the chosen rows or columns in $M(\lambda)$ can be arbitrary as long as the summation of (14) and (15) equals 0. Since equation (20) corresponds to a vector Y with elements not all equal to 0, then Y corresponds to the null space of $M'(\lambda)$. From here, we can see that $M'(\lambda)$ is non-invertible, and its determinant for $M'(\lambda)$ is 0. As such, λ can be determined from the relation $\det(M'(\lambda)) = 0$. Hence, a solution of α must be a root of the function:

$$f(\lambda) = \det(M'(\lambda)). \tag{21}$$

To this end, the degree of polynomial $f(\lambda)$ is directly pertinent to the number of columns containing λ in $M'(\lambda)$. The relation $\frac{n}{2}$ gives the maximum number of columns possible. Note that n refers to the number of elements in the ciphertext, CT vector. Observe that the number of elements in a vector cannot be exponentially large for the multivariate cryptosystem to be practical. Thus, the maximum number of roots is not exponentially many. Hence, if the value $\mu + e$ is known to the adversary, then the adversary can test all possible values of α in polynomial time.

□

3.1.1 Algorithm for Proposition 3

Algorithm 7 Retrieving secret key α via Proposition 3

Input: Public key, PK and ciphertext, CT.
Output: Secret key, α.

1. For $i \in \{1, \ldots, n\}$, $a_1 \in \{0, \ldots, k-1\}$, $a_2 \in \{0, \ldots, k-1\}, \ldots, a_\beta \in \{0, \ldots, k-1\}$, construct matrix $M(\lambda)$:
2. Compute first half column using $M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = (CT_i - \lambda \cdot PK_i) \cdot x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta}$.
3. Compute second half column using $M(\lambda)_{i,a_1,a_2,\ldots,a_\beta} = -x_{1,i}^{a_1} \cdot x_{2,i}^{a_2} \cdot \ldots \cdot x_{\beta,i}^{a_\beta}$.
4. Get $[r, s] = M(\lambda)$ where r represents as number of rows and s represents as number of columns for matrix $M(\lambda)$.
5. For the case $r = s$, let $M'(\lambda) = M(\lambda)$.
6. For the case $r > s$, let $M'(\lambda)$ be an $s \times s$ sub-square matrix in $M(\lambda)$.
7. For the case $r < s$, let $M'(\lambda)$ be an $r \times r$ sub-square matrix in $M(\lambda)$.
8. Compute determinant, $\det(M'(\lambda))$.
9. Solve $f(\lambda) = \det(M'(\lambda)) = 0$.
10. List all roots of $f(\lambda)$ which contains all possible candidates of the secret key α.

3.1.2 Numerical Illustration of Proposition 3

This section shows a numerical illustration of Proposition 3 on a three variables PRP. Given $n = 16$, $k = 2$, $w = 1$, $W = 8$, $x = (4, 3, 2, 1, 2, 1, 2, 3, 4, 3, 5, 7, 9, 6, 8, 10)$, $y = (1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8)$, $z = (2, 4, 6, 8, 10, 1, 3, 5, 7, 9, 2, 3, 6, 7, 3, 7)$ and in $\mathbb{F}_{17}$. The private polynomial is given by,

$$p(x, y, z) = xyz + 3xy + 2xz + 4yz + 4x + y + 2z + 4.$$

The big error vector, E is given by,

$$E = (1, 2, 3, 4, 1, 2, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0).$$

The public key is:

$$PK = C + E$$

where $C = ev(p(x, y, z))$. We compute C as follows:

$$p(4, 1, 2) = 1,\ p(3, 1, 4) = 1,\ p(2, 2, 6) = 15,\ p(1, 2, 8) = 9,$$

$$p(2, 3, 10) = 1,\ p(1, 3, 1) = 5,\ p(2, 4, 3) = 11,\ p(3, 4, 5) = 15,$$

$$p(4, 5, 7) = 10,\ p(3, 5, 9) = 11,\ p(5, 6, 2) = 14,\ p(7, 6, 3) = 2,$$

$$p(9, 7, 6) = 1,\ p(6, 7, 7) = 1,\ p(8, 8, 3) = 0,\ p(10, 8, 7) = 6.$$

Hence, $C = (1, 1, 15, 9, 1, 5, 11, 15, 10, 11, 14, 2, 1, 1, 0, 6)$. Then, compute PK where

$$\begin{aligned} PK &= C + E \\ &= (1, 1, 15, 9, 1, 5, 11, 15, 10, 11, 14, 2, 1, 1, 0, 6) + (1, 2, 3, 4, 1, 2, 3, 4, 0, 0, 0, 0, 0, \\ &\quad 0, 0, 0) \\ &= (2, 3, 1, 13, 2, 7, 14, 2, 10, 11, 14, 2, 1, 1, 0, 6). \end{aligned}$$

A message polynomial $\mu(x, y, z) = 3y + 2z + 1$ is encoded into codeword μ where $\mu = ev(\mu(x, y, z))$. That is,

$$\mu(4, 1, 2) = 8,\ \mu(3, 1, 4) = 12,\ \mu(2, 2, 6) = 2,\ \mu(1, 2, 8) = 6,$$

$$\mu(2, 3, 10) = 13,\ \mu(1, 3, 1) = 12,\ \mu(2, 4, 3) = 2,\ \mu(3, 4, 5) = 6,$$

$$\mu(4, 5, 7) = 13,\ \mu(3, 5, 9) = 0,\ \mu(5, 6, 2) = 6,\ \mu(7, 6, 3) = 8,$$

$$\mu(9, 7, 6) = 0,\ \mu(6, 7, 7) = 2,\ \mu(8, 8, 3) = 14,\ \mu(10, 8, 7) = 5.$$

Therefore, we have

$$\mu = (8, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5).$$

Let $\alpha = 3 \in \mathbb{F}_{17}$ be the private constant and a small error vector, e is given by,

$$e = (3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$

of weight $w = 1$. The ciphertext CT is:

$$\begin{aligned}
CT &= \mu + \alpha \times PK + e \\
&= (8,12,2,6,13,12,2,6,13,0,6,8,0,2,14,5) + 3 \times (2,3,1,13,2,7,14,2,10,11, \\
&\quad 14,2,1,1,0,6) + (3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
&= (8,12,2,6,13,12,2,6,13,0,6,8,0,2,14,5) + (6,9,3,5,6,4,8,6,13,16,8,6,3, \\
&\quad 3,0,1) + (3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\
&= (0,4,5,11,2,16,10,12,9,16,14,14,3,5,14,6).
\end{aligned}$$

Next, proceed to attack the ciphertext, CT. Let $M(\lambda)$ be the matrix of the system:

A1. $M(\lambda)_{i,a,b,c} = (CT_i - \lambda \cdot PK_i) \cdot (x_i)^a \cdot (y_i)^b \cdot (z_i)^c$

A2. $M(\lambda)_{i,a,b,c} = -(x_i)^a \cdot (y_i)^b \cdot (z_i)^c$

where $i \in \{1,\ldots,16\}$, $a \in \{0,1\}$, $b \in \{0,1\}$ and $c \in \{0,1\}$ for $(A1)$ and $(A2)$. For the first half column of matrix $M(\lambda)$ we use $(A1)$. Hence, when $i = 1$, $a = 0$, $b = 0$ and $c = 0$ then,

$$\begin{aligned}
M(\lambda)_{1,0,0,0} &= (CT_1 - \lambda \cdot PK_1) \cdot (x_1)^0 \cdot (y_1)^0 \cdot (z_1)^0 \\
&= (0 - \lambda \cdot 2) \cdot 4^0 \cdot 1^0 \cdot 2^0 \\
&= -2\lambda.
\end{aligned}$$

When $i = 5$, $a = 1$,$b = 1$ and $c = 1$ then,

$$\begin{aligned}
M(\lambda)_{5,1,1,1} &= (CT_5 - \lambda \cdot PK_5) \cdot (x_5)^1 \cdot (y_5)^1 \cdot (z_5)^1 \\
&= (2 - \lambda \cdot 2) \cdot 2 \cdot 3 \cdot 10 \\
&= 1 - \lambda.
\end{aligned}$$

When $i = 3$, $a = 0$, $b = 1$ and $c = 0$ then,

$$\begin{aligned}
M(\lambda)_{3,0,1,0} &= (CT_3 - \lambda \cdot PK_3) \cdot (x_3)^0 \cdot (y_3)^1 \cdot (z_3)^0 \\
&= (5 - \lambda \cdot 1) \cdot 2^0 \cdot 2^1 \cdot 6^0 \\
&= 10 - 2\lambda.
\end{aligned}$$

For the second half column of matrix $M(\lambda)$ we use $(A2)$. When $i = 2$, $a = 1$, $b = 1$ and $c = 1$ then,

$$\begin{aligned}
M(\lambda)_{2,1,1,1} &= -(x_2)^1 \cdot (y_2)^1 \cdot (z_2)^1 \\
&= -(3^1) \cdot (1^1) \cdot (4^1) \\
&= 5.
\end{aligned}$$

When $i = 3$, $a = 0$, $b = 0$ and $c = 1$ then,

$$\begin{aligned}
M(\lambda)_{3,0,0,1} &= -(x_3)^0 \cdot (y_3)^0 \cdot (z_3)^1 \\
&= -(2^0) \cdot (2^0) \cdot (6^1) \\
&= 11.
\end{aligned}$$

When all the entries in $M(\lambda)$ are calculated then we got $M(\lambda)$ in Appendix B. As we can see from Appendix B, the dimension for $M(\lambda)$ is 16×16. Then consider $M(\lambda)$ with $\lambda = 0$ and compute the rank of the matrix $M(0)$ by using Gaussian elimination, and we obtain the rank equal to 16. Since $M(\lambda)$ is a square matrix and the rank $M(0) = 16$ then we take $M(\lambda)$ to be $M'(\lambda)$. Next, compute determinant $f(\lambda)$,

$$\begin{aligned} f(\lambda) &= \det\left(M'(\lambda)\right) \\ &= 271882418359124720\lambda^8 - 96606074550855504\lambda^7 + 41785385885462112\lambda^6 \\ &\quad + 72035251542097952\lambda^5 - 93483385807761678\lambda^4 \\ &\quad - 83609531226382208\lambda^3 + 531102247352419712\lambda^2 \\ &\quad - 271403260445379632\lambda + 117960521373046160. \end{aligned}$$

The highest degree of $f(\lambda)$ is 8. This coincides with that $M'(\lambda)$ has 8 columns containing λ. Upon factoring $f(\lambda)$ modulo $q = 11$ we obtain the following:

$$f(\lambda) = 3\lambda(\lambda^6 + 11\lambda^5 + \lambda^4 + \lambda^3 + 10\lambda^2 + 5\lambda + 14)(\lambda - 3).$$

From here, we obtain 1 root, which is $\lambda = 3$. Inline with Proposition 3, $M'(3)$ is indeed a non-invertible matrix. Next, nullspace, Y is calculated by using $\lambda = 3$. Observe the root $\lambda = 3$ will result in

$$Y = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 6 \\ 0 \\ 0 \\ 0 \\ 10 \\ 3 \\ 13 \\ 0 \\ 6 \\ 12 \\ 1 \\ 0 \end{bmatrix}.$$

This shows that $2^{\beta-1}k^2 = 16$ and Y corresponds to

$$Y^T = (v_0, v_1, \ldots, v_{15}, n_0, n_1, \ldots, n_{15}).$$

3.2 Indistinguishable under Chosen Plaintext Attack on the AAK–Cryptosystem in Multivariate Setting

This section shows that the AAK–Cryptosystem in multivariate setting is not IND-CPA secure. Observe within the multivariate PRP cryptosystem that the weight of a small error vector, e, must be $w < \frac{n-k}{2}$ which means there are $n - w$ elements equal to 0 in a small error vector, e. Therefore, we can apply this fact to show that the multivariate PRP cryptosystem is not IND-CPA secure. The proposition for this attack is as follows,

Proposition 4. *If vector $\mu + e$ has been obtained, then multivariate PRP cryptosystem is not IND-CPA secured.*

Proof. PPTA will do the following:

1. Choose two messages, μ_0, μ_1 in which identical elements do not share the same position in the vector and send it to the random oracle.
2. Random oracle relays back the ciphertext where $CT = \mu_b + \alpha \times PK + e$.
3. Compute α based on Proposition 3.
4. Compute $CT - \alpha \times PK = \mu_b + e$.
5. Since PPTA knows about secret key, α, then PPTA can check $\mu_b + e$ vector entry positions. From the fact that e has vector elements equal to 0 totaling $n - w$, then PPTA can identify b.

Note that if the PPTA chooses a wrong root from $f(\lambda)$, it will yield to a wrong value of α. As such, $CT - \alpha \times PK$ would result in a meaningless vector to compare with either μ_0 or μ_1. The adversary eventually chooses the next root available. This process is feasible since the number of roots is not exponentially many. From here, we can see that multivariate PRP cryptosystem is not IND-CPA because PPTA can identify which vector μ_b was utilized to be encrypted with $Pr(b' = b) = 1$. Furthermore, on a side note, PPTA can also identify vector e. □

3.2.1 Algorithm for Proposition 4

We present our Algorithm for Proposition 4 on proving that multivariate PRP cryptosystem is not IND-CPA secure.

Algorithm 8 IND-CPA on multivariate PRP cryptosystem via Proposition 4

Input: Messages pair (μ_0, μ_1)
Output: b where $b \in \{0, 1\}$

1. PPTA chooses 2 messages, (μ_0, μ_1) where identical elements do not share the same position in the vectors.
2. PPTA transmits 2 messages to random oracle.
3. Random oracle chooses 1 message between (μ_0, μ_1).
4. Random oracle encrypts the message and outputs $CT = \mu_b + \alpha \times PK + e$.
5. PPTA calculates α.
6. PPTA calculates $CT - \alpha \times PK = \mu_b + e$.
7. PPTA check $\mu_b + e$ with (μ_0, μ_1) to identify b.

3.2.2 A Three Variables Numerical Illustration of Proposition 4

This section presents a numerical illustration of IND-CPA on a three variables PRP cryptosystem based on Proposition 4. Given $n = 16$, $x = (4, 3, 2, 1, 2, 1, 2, 3, 4, 3, 5, 7, 9, 6, 8, 10)$, $y = (1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8)$ and $z = (2, 4, 6, 8, 10, 1, 3, 5, 7, 9, 2, 3, 6, 7, 3, 7)$ in $\mathbb{F}_{17}$. The private polynomial is given by,

$$p(x, y, z) = xyz + 3xy + 2xz + 4yz + 4x + y + 2z + 4.$$

The big error vector, E is given by,

$$E = (1, 2, 3, 4, 1, 2, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0).$$

The public key is:

$$PK = C + E$$

where $C = ev(p(x, y, z))$ hence,

$$\begin{gathered} p(4,1,2) = 1,\ p(3,1,4) = 1,\ p(2,2,6) = 15,\ p(1,2,8) = 9, \\ p(2,3,10) = 1,\ p(1,3,1) = 5,\ p(2,4,3) = 11,\ p(3,4,5) = 15, \\ p(4,5,7) = 10,\ p(3,5,9) = 11,\ p(5,6,2) = 14,\ p(7,6,3) = 2, \\ p(9,7,6) = 1,\ p(6,7,7) = 1,\ p(8,8,3) = 0,\ p(10,8,7) = 6. \end{gathered}$$

Therefore,

$$\begin{aligned} PK &= C + E \\ &= (1,1,15,9,1,5,11,15,10,11,14,2,1,1,0,6) + (1,2,3,4,1,2,3,4,0,0,0,0,0, \\ &\quad 0,0,0) \\ &= (2,3,1,13,2,7,14,2,10,11,14,2,1,1,0,6). \end{aligned}$$

PPTA chooses two messages which are $\mu_0(x, y, z) = 3y+2z+1$ and $\mu_1(x, y, z) = y+8z+5$. These two messages are encoded into codeword μ_0 and μ_1 respectively where $\mu_b = ev(\mu(x, y, z))$ for $b \in \{0, 1\}$. For $\mu_0(x, y, z) = xy + 2x + 4y + 3$, it is encoded as follows,

$$\begin{gathered} \mu_0(4,1,2) = 8,\ \mu_0(3,1,4) = 12,\ \mu_0(2,2,6) = 2,\ \mu_0(1,2,8) = 6, \\ \mu_0(2,3,10) = 13,\ \mu_0(1,3,1) = 12,\ \mu_0(2,4,3) = 2,\ \mu_0(3,4,5) = 6, \\ \mu_0(4,5,7) = 13,\ \mu_0(3,5,9) = 0,\ \mu_0(5,6,2) = 6,\ \mu_0(7,6,3) = 8, \\ \mu_0(9,7,6) = 0,\ \mu_0(6,7,7) = 2,\ \mu_0(8,8,3) = 14,\ \mu_0(10,8,7) = 5. \end{gathered}$$

Then, $\mu_0 = (8, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5)$ is obtained. For $\mu_1(x, y, z) = y + 8z + 5$, it is encoded as follows,

$$\begin{gathered} \mu_1(4,1,2) = 5,\ \mu_1(3,1,4) = 4,\ \mu_1(2,2,6) = 4,\ \mu_1(1,2,8) = 3, \\ \mu_1(2,3,10) = 3,\ \mu_1(1,3,1) = 16,\ \mu_1(2,4,3) = 16,\ \mu_1(3,4,5) = 15, \\ \mu_1(4,5,7) = 15,\ \mu_1(3,5,9) = 14,\ \mu_1(5,6,2) = 10,\ \mu_1(7,6,3) = 1, \\ \mu_1(9,7,6) = 9,\ \mu_1(6,7,7) = 0,\ \mu_1(8,8,3) = 3,\ \mu_1(10,8,7) = 1. \end{gathered}$$

Then, we get $\mu_1 = (5, 4, 4, 3, 3, 16, 16, 15, 15, 14, 10, 1, 9, 0, 3, 1)$. The PPTA must ensure that identical elements in the 2 message vectors must not share the same location. Next, the 2 message vectors (μ_0, μ_1) will send by PPTA to random oracle. The random oracle will pick one of the message vectors and encrypt it, and publishes CT where

$$\begin{aligned} CT &= \mu_b + \alpha \times PK + e \\ &= (8,12,2,6,13,12,2,6,13,0,6,8,0,2,14,5) + 3 \times (2,3,1,13,2,7,14,2,10,11, \\ &\quad 14,2,1,1,0,6) + (3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\ &= (8,12,2,6,13,12,2,6,13,0,6,8,0,2,14,5) + (6,9,3,5,6,4,8,6,13,16,8,6,3, \\ &\quad 3,0,1) + (3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0) \\ &= (0,4,5,11,2,16,10,12,9,16,14,14,3,5,14,6). \end{aligned}$$

Since the value of the secret key, α, can be computed based on Proposition 3, then PPTA will retrieve $\alpha = 3$. Next, PPTA calculates the following equation,

$$CT - \alpha \times PK = \mu_b + e$$

and obtains $\mu_b + e = (11, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5)$. The PPTA can check the entry positions vis-a-vis equation $CT - \alpha \times PK$. Therefore, PPTA can finally identify b from $\mu_b + e$ because vector e has $n - w$ elements equal to 0. To this end, the PPTA can identify $b' = 0$ with a probability equal to one.

Remark 3. We demonstrate numerically in this Remark why this fact leads to the conclusion that the multivariate cryptosystem is not IND-CPA secure. First, the adversary will choose two message polynomials which are $\mu_0(x, y, z) = 3y + 2z + 1$ and $\mu_1(x, y, z) = y + 8z + 5$. Then, adversary encodes the two messages into codewords where $\mu_0 = (8, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5)$ and $\mu_1 = (5, 4, 4, 3, 3, 16, 16, 15, 15, 14, 10, 1, 9, 0, 3, 1)$ and transfers the two messages to random oracle. Next, the random oracle will encrypt one of the messages and send ciphertext, CT. With the obtained value of $\lambda = 3$, the adversary can compute the following equation,

$$\begin{aligned} CT - 3 \times PK &= (0, 4, 5, 11, 2, 16, 10, 12, 9, 16, 14, 14, 3, 5, 14, 6) - 3\times \\ &\quad (2, 3, 1, 13, 2, 7, 14, 2, 10, 11, 14, 2, 1, 1, 0, 6) \\ &= (11, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5). \end{aligned} \tag{22}$$

This shows that $CT - \alpha \times PK = \mu + e$ where,

$$\begin{aligned} \mu + e &= (8, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5) + (3, 0, 0, 0, 0, 0, 0, 0, \\ &\quad 0, 0, 0, 0, 0, 0, 0, 0) \\ &= (11, 12, 2, 6, 13, 12, 2, 6, 13, 0, 6, 8, 0, 2, 14, 5). \end{aligned} \tag{23}$$

The $\mu + e$ is being compared with μ_0 and μ_1 to determine which message has been encrypted and adversary also can identify which elements in $\mu + e$ contains errors. As we can see here, μ_0 is the message that used to be encrypted since $\mu + e$ has many elements that are similar to μ_0.

4 Conclusion

This paper presents an extended version of the AAK–Cryptosystem in multivariate setting. Based on the analysis above, a set of λ is obtained from the determinant $f(\lambda)$, which contains the correct value of α. Since the weight of a small error vector, e must be $\frac{n-k}{2}$, which gives us information about the zero elements in e. Hence, this shows that the AAK–Cryptosystem in multivariate setting is not IND-CPA secure.

Acknowledgments

The research was supported by Ministry of Education of Malaysia with Fundamental Research Grant Scheme (FRGS/1/2019/STG06/UPM/02/8). It is also partially supported by Mediterranea Universiti of Reggio Calabria (UNIRC) Research Grant (UPM/INSPEM/7003/1/GERANANTARABANGSA/638007312210065).

A Matrix $M(\lambda)$

$$M(\lambda)^T = \begin{bmatrix} (CT_1 - \lambda \cdot PK_1) & (CT_2 - \lambda \cdot PK_2) & \cdots & (CT_n - \lambda \cdot PK_n) \\ (CT_1 - \lambda \cdot PK_1) \cdot (x_{1,1}) & (CT_2 - \lambda \cdot PK_2) \cdot (x_{1,2}) & \cdots & (CT_n - \lambda \cdot PK_n) \cdot (x_{1,n}) \\ (CT_1 - \lambda \cdot PK_1) \cdot (x_{2,1}) & (CT_2 - \lambda \cdot PK_2) \cdot (x_{2,2}) & \cdots & (CT_n - \lambda \cdot PK_n) \cdot (x_{2,n}) \\ \vdots & \vdots & \vdots & \vdots \\ (CT_1 - \lambda \cdot PK_1) \cdot (x_{\beta,1}) & (CT_2 - \lambda \cdot PK_2) \cdot (x_{\beta,2}) & \cdots & (CT_n - \lambda \cdot PK_n) \cdot (x_{\beta,n}) \\ \vdots & \vdots & \vdots & \vdots \\ (CT_1 - \lambda \cdot PK_1) \cdot x_{1,1}^{k-1} \cdot x_{2,1}^{k-1} \cdot \ldots \cdot x_{\beta,1}^{k-1} & (CT_2 - \lambda \cdot PK_2) \cdot x_{1,2}^{k-1} \cdot x_{2,2}^{k-1} \cdot \ldots \cdot x_{\beta,2}^{k-1} & \cdots & (CT_n - \lambda \cdot PK_n) \cdot x_{1,n}^{k-1} \cdot x_{2,n}^{k-1} \cdot \ldots \cdot x_{\beta,n}^{k-1} \\ -1 & -1 & \cdots & -1 \\ -x_{1,1} & -x_{1,2} & \cdots & -x_{1,n} \\ -x_{2,1} & -x_{2,2} & \cdots & -x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ -x_{\beta,1} & -x_{\beta,2} & \cdots & -x_{\beta,n} \\ \vdots & \vdots & \vdots & \vdots \\ -x_{1,1}^{k-1} \cdot x_{2,1}^{k-1} \cdot \ldots \cdot x_{\beta,1}^{k-1} & -x_{1,2}^{k-1} \cdot x_{2,2}^{k-1} \cdot \ldots \cdot x_{\beta,2}^{k-1} & \cdots & -x_{1,n}^{k-1} \cdot x_{2,n}^{k-1} \cdot \ldots \cdot x_{\beta,n}^{k-1} \end{bmatrix}.$$

B Matrix $M(\lambda)$ for Numerical Illustration of Proposition 3

$$M(\lambda) = \begin{bmatrix}
-2\lambda & -4\lambda & -2\lambda & -4\lambda & -8\lambda & -16\lambda & -8\lambda & -16\lambda & 16 & 15 & 16 & 15 & 13 & 9 & 13 & 9 \\
4-7\lambda & 16-12\lambda & 4-3\lambda & 16-12\lambda & 12-9\lambda & 14-2\lambda & 12-8\lambda & 14-2\lambda & 16 & 13 & 16 & 13 & 14 & 5 & 14 & 5 \\
5-\lambda & 13-6\lambda & 10-2\lambda & 9-12\lambda & 10-2\lambda & 9-12\lambda & 3-14\lambda & 1-7\lambda & 16 & 11 & 15 & 5 & 15 & 5 & 13 & 10 \\
11-13\lambda & 3-2\lambda & 5-9\lambda & 6-4\lambda & 11-13\lambda & 3-2\lambda & 5-9\lambda & 6-4\lambda & 16 & 9 & 15 & 1 & 16 & 9 & 15 & 1 \\
2-2\lambda & 3-3\lambda & 6-6\lambda & 9-9\lambda & 4-4\lambda & 6-6\lambda & 12-12\lambda & 1-\lambda & 16 & 7 & 14 & 4 & 15 & 14 & 11 & 8 \\
16-7\lambda & 16-7\lambda & 14-4\lambda & 14-4\lambda & 16-7\lambda & 16-7\lambda & 14-4\lambda & 14-4\lambda & 16 & 16 & 14 & 14 & 16 & 16 & 14 & 14 \\
10-14\lambda & 13-8\lambda & 6-5\lambda & 1-15\lambda & 3-11\lambda & 9-16\lambda & 12-10\lambda & 2-13\lambda & 16 & 14 & 13 & 5 & 15 & 11 & 9 & 10 \\
12-2\lambda & 9-10\lambda & 14-8\lambda & 2-6\lambda & 2-6\lambda & 10-13\lambda & 8-7\lambda & 6-\lambda & 16 & 12 & 13 & 14 & 14 & 2 & 5 & 8 \\
9-10\lambda & 12-2\lambda & 11-16\lambda & 9-10\lambda & 2-6\lambda & 14-8\lambda & 10-13\lambda & 2-6\lambda & 16 & 10 & 12 & 16 & 13 & 6 & 14 & 13 \\
16-11\lambda & 8-15\lambda & 12-4\lambda & 6-2\lambda & 14-16\lambda & 7-8\lambda & 2-12\lambda & 1-6\lambda & 16 & 8 & 12 & 6 & 14 & 7 & 2 & 1 \\
14-14\lambda & 11-11\lambda & 16-16\lambda & 15-15\lambda & 2-2\lambda & 4-4\lambda & 12-12\lambda & 7-7\lambda & 16 & 15 & 11 & 5 & 12 & 7 & 4 & 8 \\
14-2\lambda & 8-6\lambda & 16-12\lambda & 14-2\lambda & 13-14\lambda & 5-8\lambda & 10-16\lambda & 13-14\lambda & 16 & 14 & 11 & 16 & 10 & 13 & 9 & 10 \\
3-\lambda & 1-6\lambda & 4-7\lambda & 7-8\lambda & 10-9\lambda & 9-3\lambda & 2-12\lambda & 12-4\lambda & 16 & 11 & 10 & 9 & 8 & 14 & 5 & 13 \\
5-\lambda & 1-7\lambda & 1-7\lambda & 7-15\lambda & 13-6\lambda & 6-8\lambda & 6-8\lambda & 8-5\lambda & 16 & 10 & 10 & 2 & 11 & 9 & 9 & 12 \\
14 & 8 & 10 & 13 & 10 & 13 & 12 & 2 & 16 & 14 & 9 & 10 & 9 & 10 & 4 & 12 \\
6-6\lambda & 8-8\lambda & 14-14\lambda & 13-13\lambda & 9-9\lambda & 12-12\lambda & 4-4\lambda & 11-11\lambda & 16 & 10 & 9 & 12 & 7 & 15 & 5 & 1
\end{bmatrix}.$$

References

[1] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. *IET Information Security*, 10(6):288–303, 2016.

[2] Ruma Kareem Ajeena, Hailiza Kamarulhaili, and Sattar B Almaliky. Bivariate polynomials public key encryption schemes. *International Journal of Cryptology Research*, 4(1):73–83, 2013.

[3] Daniel Augot and Matthieu Finiasz. A public key encryption scheme based on the polynomial reconstruction problem. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 229–240. Springer, 2003.

[4] Daniel Augot, Matthieu Finiasz, and Pierre Loidreau. Using the trace operator to repair the polynomial reconstruction based cryptosystem presented at eurocrypt 2003. *Cryptology ePrint Archive*, 2003.

[5] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. On quantum indistinguishability under chosen plaintext attack. *IACR Cryptol. ePrint Arch.*, 2020:596, 2020.

[6] Alvaro Cintas-Canto, Mehran Mozaffari-Kermani, Reza Azarderakhsh, and Kris Gaj. Crc-oriented error detection architectures of post-quantum cryptography niederreiter key generator on fpga. In *2022 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pages 1–7. IEEE, 2022.

[7] Jean-Sebastien Coron. Cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem. In *Public Key Cryptography–PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Proceedings 7*, pages 14–27. Springer, 2004.

[8] Philippe Gaborit, Ayoub Otmani, and Hervé Talé Kalachi. Polynomial-time key recovery attack on the faure–loidreau scheme based on gabidulin codes. *Designs, Codes and Cryptography*, 86:1391–1403, 2018.

[9] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 28–37. IEEE, 1998.

[10] NASA Jamal and Muhammad Rezal Kamel. Novel forgery mechanisms in multivariate signature schemes. *Comput. Sci*, 18(3):451–461, 2023.

[11] Stephen Jordan, Apr 2011. https://quantumalgorithmzoo.org/.

[12] Jasmin Kaur, Alvaro Cintas Canto, Mehran Mozaffari Kermani, and Reza Azarderakhsh. A comprehensive survey on the implementations, attacks, and countermeasures of the current nist lightweight cryptography standard. *arXiv preprint arXiv:2304.06222*, 2023.

[13] Mehran Mozaffari Kermani and Reza Azarderakhsh. Lightweight hardware architectures for fault diagnosis schemes of efficiently-maskable cryptographic substitution boxes. In *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 764–767. IEEE, 2016.

[14] Aggelos Kiayias and Moti Yung. Cryptanalyzing the polynomial reconstruction based public-key system under optimal parameter choice.

[15] Aggelos Kiayias and Moti Yung. Polynomial reconstruction based cryptography: (a short survey). In *International Workshop on Selected Areas in Cryptography*, pages 129–133. Springer, 2001.

[16] Aggelos Kiayias and Moti Yung. Directions in polynomial reconstruction based cryptography. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 87(5):978–985, 2004.

[17] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *2010 IEEE international symposium on information theory*, pages 2682–2685. IEEE, 2010.

[18] Cosimo Laneve, Tudor A Lascu, and Vania Sordoni. The interval analysis of multilinear expressions. *Electronic Notes in Theoretical Computer Science*, 267(2):43–53, 2010.

[19] Cheng-Yi Lin and Ja-Ling Wu. Cryptanalysis and improvement of a chaotic map-based image encryption system using both plaintext related permutation and diffusion. *Entropy*, 22(5):589, 2020.

[20] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, 1999.

[21] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[22] Sattar B Sadkhan and KH Ruma. Evaluation of polynomial reconstruction problem using lagrange interpolation method. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 1399–1403. IEEE, 2006.

[23] 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.

[24] Siti Nabilah Yusof and MR Kamel Ariffin. An empirical attack on a polynomial reconstruction problem potential cryptosystem. *Int. J. Cryptol. Res*, 11:31–48, 2021.

[25] Siti Nabilah Yusof, Muhammad Rezal Kamel Ariffin, Terry Shue Chien Lau, Nur Raidah Salim, Sook-Chin Yip, and Timothy Tzen Vun Yap. An IND-CPA analysis of a cryptosystem based on bivariate polynomial reconstruction problem. *Axioms*, 12(3):304, 2023.

[26] Shuaishuai Zhu and Yiliang Han. Generative trapdoors for public key cryptography based on automatic entropy optimization. *China Communications*, 18(8):35–46, 2021.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 97–109.

Related-Key Boomerang Attack on Mini-AES

Ferdinan Setyo Puji[1] and Santi Indarjani[1,2]

[1] National Cyber and Crypto Polytechnic, Bogor, Indonesia
ferdinan.setyo@student.poltekssn.ac.id
[2] National Cyber and Crypto Polytechnic, Bogor, Indonesia santi.indarjani@poltekssn.ac.id

Abstract. The Boomerang attack is a variant of differential attack conducted by utilizing two independent short differential characteristics with high probability which are can-not be combined as a longer characteristic. Related key boomerang attack applied under assumption that the key used for encryption and decryption are relating, which can be manipulate to find the distinguisher. This paper showed the implementation of this attack on Mini-AES as an educational mean to give an understanding how the related-key boomerang attack works. This paper focus on determining the right quartet as a distinguisher bases on samples. By defining a relation among the keys this attack builds a related-key boomerang path to identify potential weaknesses in Mini-AES security. By carrying out differential analysis using 2^8 data complexity on Mini-AES, this attack succeeded in obtaining 48 correct quartet pairs. This attack provides critical insight into cryptographic variants that are vulnerable to similar attacks. This research provides an in-depth understanding of this attack and its relevance in testing the security of Mini-AES cryptographic systems.

Keywords: Boomerang attack · Cryptanalysis · MiniAES · Related-Key

1 Introduction

The Mini Advanced Encryption Standard (Mini-AES) proposed by Phan[1] serves as a valuable testbed for students of cryptanalysis, providing a platform for practical exploration and analysis of the Advanced Encryption Standard (AES). It facilitates hands-on experience in understanding the complexity of the block cipher and the special characteristics of the AES algorithm. By offering a simplified version of AES, Mini-AES allows students to gain insight into the basic principles of AES encryption and its vulnerability to cryptanalysis techniques. Boomerang attacks are a significant cryptographic technique that has been widely studied and applied in the context of various encryption algorithms. First proposed by Wagner in 1999 [2], boomerang attacks have evolved since then, with variants and improved techniques developed to exploit vulnerabilities in cryptographic systems. Most notably, boomerang redirection techniques were introduced to minimize the number of active S-boxes in the key schedule. Boomerang attacks are a variant of Differential Cryptanalysis, where the ciphertext is considered as a series of two sub- ciphers. This attack works using two short independence differentials with high probability which are applied on each sub-cipher. These differentials are combined in an adaptively chosen plaintext and ciphertext attack to exploit the property of ciphertexts having high probability [6]. Related-key attacks are attacks that exploit the relationship between keys to compromise the reliability of encryption algorithms [3]. This attack has been studied for intensive way in the context of various ciphertext blocks, including AES-192 and AES-256. Biryukov & Khovratovich [4] developed the first related-key attack on the full AES-192, demonstrating its vulnerability to this widely used encryption standard. Related-key attacks apply

differential cryptanalysis to ciphertexts that use different but related keys and consider the information that can be extracted from encryption under these keys [7]. Ciphertexts with weak key schedules are vulnerable to this type of attack.

This paper aims to show the implementation of a related key-attack on a testbed algorithm Mini-AES as a target, to give a simple illustration for an educational purpose. Hopefully this paper can provide a good example to understanding how this attack works. The research is limited on the search of related key boomerang distinguisher.

2 Theoretical Background

2.1 Mini Advanced Encryption Algorithm

The description Mini Advanced Encryption Standard (Mini-AES) was developed by Raphael Chung-Wei Phan [1] as a simplified version of the AES, primarily for educational purposes which is reffered in this section. The block size is 16-bits with input key also 16-bit length. As in AES, the encryption process also involves 4 functions as are described in 2.1.1. The scheme of Mini-AES is shown on Figure 1.

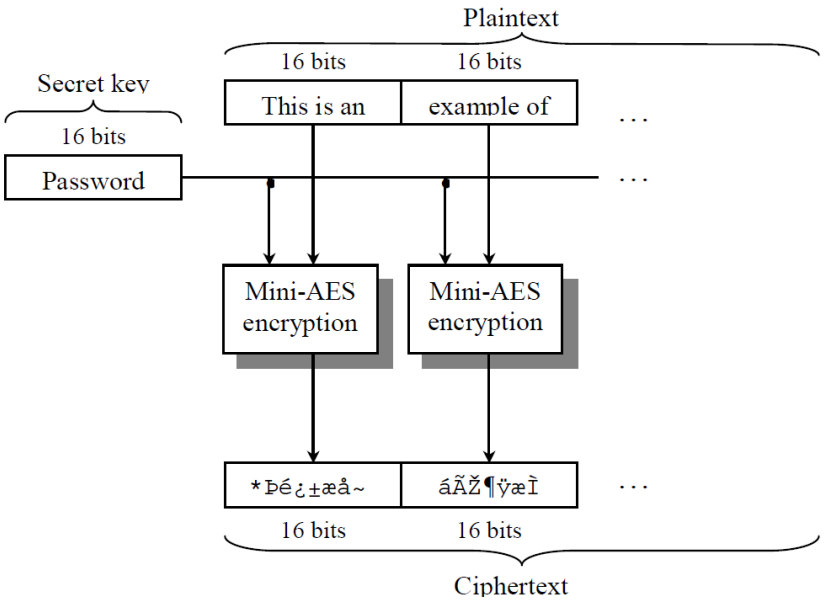


Figure 1: Mini-AES Scheme

2.1.1 Components of Mini-AES

There are 4 components used in Mini-AES encryption/decryption scheme. The message P will be partitioned into blocks of 16-bits that notated as: $P = (p_0, p_1, p_2, p_3)$ which is represented as a matrix with 2 rows and 2 columns of 4 bits (a nibble).

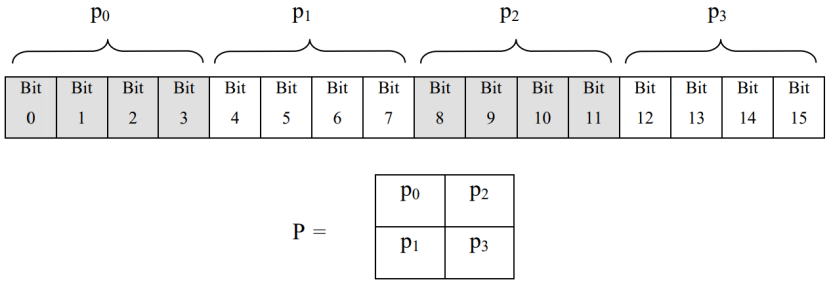


Figure 2: Matrix Representation of the 16-bit Block

2.1.1.1 NibbleSub, γ

The NibbleSub operation is a simple operation that replaces each input nibble with an output nibble according to a 4×4 substitution table (S-box), as given in the table in Figure 3. The S-box used in Mini AES is adopted from the first S-box in DES as can be seen in Figure 3.

Input	Output	Input	Output
0000	1110	1000	0011
0001	0100	1001	1010
0010	1101	1010	0110
0011	0001	1011	1100
0100	0010	1100	0101
0101	1111	1101	1001
0110	1011	1110	0000
0111	1000	1111	0111

Figure 3: S-Box of Mini-AES

NibbleSub operation where $A = (a_0, a_1, a_2, a_3)$ constitutes the input block and $B = (b_0, b_1, b_2, b_3)$ constitutes the output.

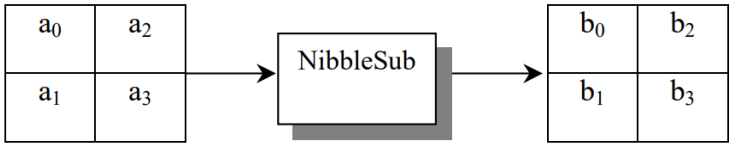


Figure 4: The Nibble Sub Operation

2.1.1.2 ShiftRow, π

The ShiftRow operation performs a left rotation on each row of the input block by a different number of nibbles. The first row remains unchanged, while the second row is rotated left by one nibble. This is illustrated in the Figure 5, where $B = (b_0, b_1, b_2, b_3)$ and $C = (c_0, c_1, c_2, c_3)$ represents the input and output blocks respectively.

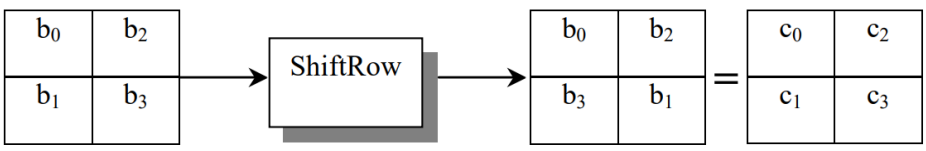


Figure 5: The ShiftRow Operation

2.1.1.3 MixColumn, θ

The MixColumn operation takes each column of the input block and multiplies it by a constant matrix to get a new output column, as given in the figure below. $C = (c_0, c_1, c_2, c_3)$ and $D = (d_0, d_1, d_2, d_3)$ denote the input and output blocks respectively.

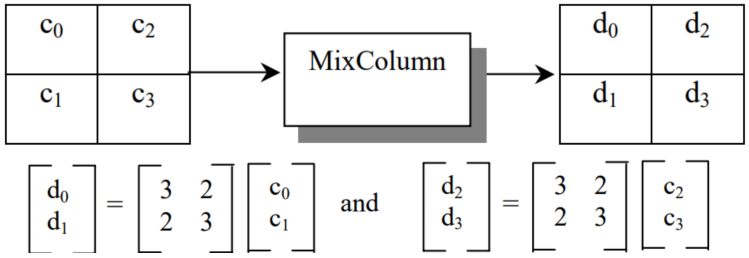


Figure 6: The MixColomn Operation

2.1.1.4 KeyAddition, σK_i

The KeyAddition operation causes each bit of the input block, $D = (d_0, d_1, d_2, d_3)$, to be exclusively XORed with the corresponding bit of the i-th round key, $K_i = (k_0, k_1, k_2, k_3)$ to get a 16-bit long output block, $E = (e_0, e_1, e_2, e_3)$, as shown in the figure below the round key is derived from the key, K, by using the key schedule. For each bit, the exclusive-OR operation causes the output bit to be '1' if the corresponding bits of the input block and the round key are different. Otherwise, the output bit becomes '0' if the two bits are the same.

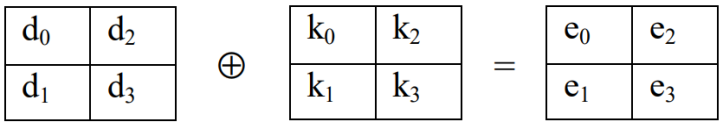


Figure 7: The KeyAddition Operation

2.1.1.5 key Schedule

In Mini-AES, the 16-bit secret key goes through the key schedule algorithm to generate one 16-bit round key, K0, to be used before the first round, and one 16-bit round key, K_2 and K_3, to be used in each round of Mini-AES. Mini-AES encryption is defined to have 2 rounds, so three round keys are needed as shown in Table 1.

2.1.2 Mini-AES Encryption and Decryption.

The application of the four components Nibble-Sub, Shift-Row, Mix-Column, and Key-Addition together forms one round. Full Mini-AES encryption consists of two such rounds, with the exception of Mix-Column from the last round and the addition of an extra Key-Addition before the first round. The encryption process can be seen in Figure 8.

Table 1: Key Schedule of Mini-AES

Putaran	Nilai Kunci
0	$w_0 = k_0$ $w_1 = k_1$ $w_2 = k_2$ $w_3 = k_3$
1	$w_4 = w_0 \oplus \text{NibbleSub}(w_3) \oplus \text{rcon}(1)$ $w_5 = w_1 \oplus w_4$ $w_6 = w_2 \oplus w_5$ $w_7 = w_3 \oplus w_6$
2	$w_8 = w_4 \oplus \text{NibbleSub}(w_7) \oplus \text{rcon}(2)$ $w_9 = w_5 \oplus w_8$ $w_{10} = w_6 \oplus w_9$ $w_{11} = w_7 \oplus w_{10}$

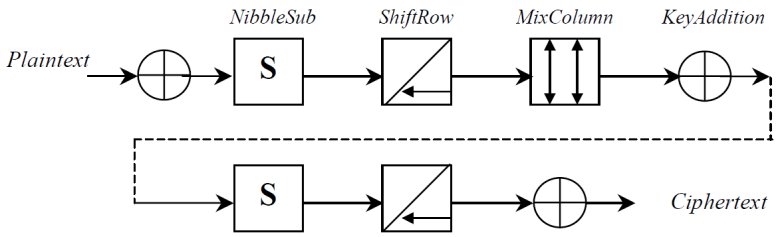


Figure 8: The Mini-AES Encryption Process

The encryption can be represented as follow:

$$\text{Mini-AES encryption} = \sigma K_2 \circ \pi \circ \gamma \circ \sigma K_1 \circ \theta \circ \pi \circ \gamma \circ \sigma K_0$$

Where the symbol $\circ$ refers to function composition and the execution order is from right to left. On the other hand, to retrieve the original plain text, the reverse process of encryption must be performed on the ciphertext that is called as decryption. The decryption is the reverse of encryption, where the subkey will be processed in a reversed order and the also the Nibble-Sub. The inverse s-box of mini-AES then is computed by inverting the input and output as is shown in Figure 9.

Input	Output		Input	Output
0000	1110		1000	0111
0001	0011		1001	1101
0010	0100		1010	1001
0011	1000		1011	0110
0100	0001		1100	1011
0101	1100		1101	0010
0110	1010		1110	0000
0111	1111		1111	0101

Figure 9: Inverse Sbox of Mini-AES

2.2 Related-Key Boomerang Attack

The related-key boomerang attack on AES, as described by Michael Gorski and Stefan Lucks [5], is a cryptanalytic technique targeting reduced-round versions of the AES-192 cipher. First, Eli Biham *et.al.* [8], explain there are some definitions.

Definition 1: Let P dan P' are two bit sequences of the same length. The bit-wise XOR operation of P and P', $P \oplus P'$, is called the difference of P, P'. Let a be the difference of the known bits and * being the difference of the unknown bits.

Definition 2: $\alpha \rightarrow \beta$ is called differential if α is the difference of plaintext $P \oplus P'$ before a non-linear operation $f(\cdot)$ and β is the difference after applying the operation, i.e. $f(P) \oplus f(P')$. The probability p associated with a differential states that the difference α turns into the difference β with probability p. The reverse direction, i.e.$\alpha \leftarrow \beta$, has probability $\hat{p}$.

During the search of related-key boomerang distinguisher step, the cipher text is treated as a series of two sub-ciphers $EK(P) = E_1K(P) \circ E_0K(P)$, where K is the key used for encryption and decryption. It is assumed that a differential related-key $\alpha \rightarrow \beta$ for E_0 occurs with probability p, while differential related-key $\gamma \rightarrow \delta$ for E_1 occurs with probability q, where α, β, γ, and δ are text differences. The backward direction E_0^{-1} and E_1^{-1} of the differential related-key for E_0 and E_1 are denoted by $\alpha \leftarrow \beta$ and $\gamma \leftarrow \delta$, and occur with probability $\hat{p}$ and $\hat{q}$ respectively. Related-key boomerang distinguisher involve four unknown but related keys, namely $Ka, Kb = Ka \oplus \Delta K^*, Kc = Ka \oplus \Delta K'$, and $Kd = Ka \oplus \Delta K^* \oplus \Delta K'$, where ΔK^* and $\Delta K'$ are the known. The attack works as follows:

1. Randomly select a set of s plaintexts $P_i, i \in \{1, ..., s\}$ and compute a set of other plaintexts P_i' such that $P_i' = P_i \oplus \alpha$ to ensure that $P_i \oplus P_i' = \alpha$.

2. Encrypt P_i under Ka, i.e. $C_i = EKa(P_i)$ and encrypt P_i' under Kb, i.e. $C_i' = EKb(P_i')$.

3. Calculate the new ciphertext $D_i = C_i \oplus \delta$ and $D_i' = C_i' \oplus \delta$.

4. Perform decryption of D_i under Kc, i.e. $Oi = E^{-1}Kc(D_i)$ and request decryption of D_i' under Kd, i.e. $O_i' = E^{-1}Kd(D_i')$. For each pair (Oi, O_i'), $i, j \in \{1, ..., s\}$

5. If $Oi \oplus O_i'$ is equal to α, keep the quartet (P_i, P_i', Oi, O_i') in the set M.

A pair $(P_i, P_j'), i, j \in \{1, ..., s\}$ with difference α satisfies the differential $\alpha \rightarrow \beta$ with probability p. The outputs of E_0 are A_i and A_j', i.e., $E_0K_a(P_i) = A_i$ and $E_0K_b(P_j') = A_j'$ having a certain difference $\beta = A_i \oplus A_j'$ with probability p.

Using two ciphertexts C_i and C_j', new ciphertexts D_i and D_j' are generated such that $D_i = C_i \oplus \delta$ and $D_j' = C_j' \oplus \delta$. Next B_i and B_j' are determined by applying decryption process on D_i and D_j' using $E_1^{-1}K_i, i \in \{c, d\}$ such that $B_i = E_1^{-1}K_c(D_i)$ and $B_j' = E_1^{-1}K_d(D_j')$. After passing through $E_1^{-1}K_i$, difference value δ should be mapped into difference value γ The difference δ turns into a difference γ with probability $\hat{q}$. Since $\delta = C_i \oplus D_i$ and $\delta = C_j' \oplus D_j'$ It is known that $\gamma = A_i \oplus B_i$ and $\gamma = A_j' \oplus B_j'$ with probability $\hat{q}^2$. Since it is known that $A_i \oplus A_j' = \beta$ with probability p, then $(A_i \oplus B_i) \oplus (A_i \oplus A_j') \oplus (A_j' \oplus B_j') = \gamma \oplus \beta \oplus \gamma = \beta = (B_i \oplus B_j')$ holds with probability $p \cdot \hat{q}^2$. The difference β turns into a difference α after passing through the differential $E_0^{-1}K_i$ with probability $\hat{p}$. Thus, a plaintext pair (P_i, P_j') with $P_i \oplus P_j' = \alpha$ produces a new plaintext pair (O_i, O_j') where $O_i \oplus O_j' = \alpha$ with probability $p \cdot \hat{p} \cdot \hat{q}^2$. A quartet containing these two pairs is defined as:

$$\text{Quartet} = \{(P_i, P_j', O_i, O_j') \mid P_i \oplus P_j' = \alpha, O_i \oplus O_j' = \alpha\}$$

The process of finding the right quartet as a distinguisher can be seen in Figure 10. The next definition is about the right related-key boomerang quartet

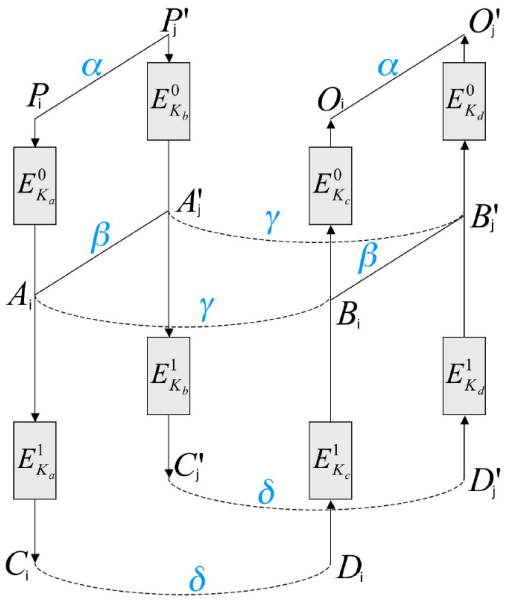


Figure 10: Boomerang Attack Scheme [8]

Definition 3. A quartet (P_i, P'_j, O_i, O'_j) satisfying

$$\begin{aligned} P_i \oplus P'_j &= \alpha = O_i \oplus O'_j, \\ A_i \oplus A'_j &= \beta = B_i \oplus B'_j, \\ A_i \oplus B_i &= \gamma = A'_j \oplus B'_j, \\ C_i \oplus D_i &= \delta = C_j \oplus D_j \end{aligned}$$

is referred to as a correct boomerang related-key quartet, which occurs with probability $P_{\text{rc}} = p \cdot \hat{p} \cdot \hat{q}^2$. A quartet (P_i, P'_j, O_i, O'_j) that only satisfies the condition $P_i \oplus P'_j = \alpha = O_i \oplus O'_j$ is called a false boomerang related-key quartet. The right related-key boomerang can be used to perform the key recovery.

The recovery attack can be performed by the attacker using the right related-key boomerang distinguishers under assumption that they do not know about the internal state of A_i, A'_j, B_i, B'_j. Let k_a, k_b, k_c, and k_d be the key bits of the last round key derived from the keys K_a, K_b, K_c, and K_d. Let $dk(C)$ be the one-round partial decryption of C under key bit k. The related-key bits as $k_b = k_a \oplus \Delta k^*$, $k_c = k_a \oplus \Delta k'$, and $k_d = k_a \oplus \Delta k^* \oplus \Delta k'$, where Δk^* and $\Delta k'$ are the differences of the last round key bits. This difference comes from the difference of the cipher keys ΔK^* and $\Delta K'$. The key recovery step works as follows:

Let M is the set of all possible related key boomerang quartets. For each key bit combination K_a:

1. Initialize the counter for each combination of key bits with zero.

   - For all quartets (P, P', O, O') that are simulated in M:

2. Perform encryption of P, P', O, O' under K_a, K_b, K_c, and K_d respectively and get the cipher quartet C, C', D, D'. Decrypt the ciphertexts C, C', D, D' under k_a, k_b, k_c, k_d, i.e. $\overline{C} = d_{k_a}(C)$, $\overline{C'} = d_{k_b}(C')$, $\overline{D} = d_{k_c}(D)$, and $\overline{D'} = d_{k_d}(D')$.

3. Test whether the difference between $\overline{C} \oplus \overline{D}$ and $\overline{C}' \oplus \overline{D'}$ has the difference expected

by the attacker depending on the related-key differential used. Increase the counter for the key candidate tested if the difference is satisfied in both pairs.

4. The key bit output k_a with the highest counter is the correct key.

There are four possibe cases at step 3, since the set M might contain both correct and incorrect boomerang related-key quartets, and the key bit combination K_a can be either correct or incorrect. A correct related-key boomerang quartet with the correct key bits will have the desired difference required to pass the test in Step 3.

3 Result and Discussion

As mentioned above, this paper only simulates the search of related-key boomerang distinguisher in order to give the knowledge how this attack is conducted. In conducting the search of related-key boomerang on Mini-AES, there are 4 steps should be performed.

First step is identifying the related keys. The process of identifying related keys is done by generating several chosen keys to find four keys K_a, K_b, K_c, and K_d that are related to each other. For example as in paper[5], the keys are related such that

$$\begin{aligned} Kb &= Ka \oplus \Delta K^*, \\ Kc &= Ka \oplus \Delta K', \\ Kd &= Ka \oplus \Delta K^* \oplus \Delta K' \end{aligned}$$

Second is identifying the key differences of K_a and K_b. In this experiment the differences are determined as follows:

1. Generate K_a and K_b that have different values at byte positions 0 and 2 (active bytes), while the values at byte positions 1 and 3 are passive bytes or bytes that have the same value between K_a and K_b.

$$\begin{aligned} K_a &= 0000\ 0000\ 0000\ 0000 \\ \Delta K^* &= 1111\ 0000\ 1111\ 0000 \\ K_b &= 1111\ 0000\ 1111\ 0000 \end{aligned}$$

2. Expanding K_a and K_b by two round

$$\begin{aligned} K_{a0} &= 0000\ 0000\ 0000\ 0000 \\ K_{a1} &= 1111\ 1111\ 1111\ 1111 \\ K_{a2} &= 1010\ 0101\ 1010\ 0101 \end{aligned}$$

$$\begin{aligned} K_{b0} &= 1111\ 0000\ 1111\ 0000 \\ K_{b1} &= 0000\ 0000\ 1111\ 1111 \\ K_{b2} &= 0101\ 0101\ 1010\ 0101 \end{aligned}$$

3. Perform XOR operation between K_a and K_b on each subkey for two rounds. So that the result of XOR K_a and K_b in each round is called key differences ΔK^*

$$\begin{aligned} \Delta K^* &= 1111\ 0000\ 1111\ 0000 \\ \Delta K_0^* &= 1111\ 0000\ 1111\ 0000 \\ \Delta K_1^* &= 1111\ 1111\ 0000\ 0000 \\ \Delta K_2^* &= 1111\ 0000\ 0000\ 0000 \end{aligned}$$

Step three is identifying the key differences of K_c and K_d. The steps to find the key differences of K_c are as follows:

1. Perform an XOR operation between K_a operation between Ka and the 16-bit characteristic $\Delta K'$, resulting in the candidate key K_c.

$$K_a = 0000\ 0000\ 0000\ 0000$$
$$\Delta K' = 1111\ 0000\ 0000\ 0000$$
$$K_c = 1111\ 0000\ 0000\ 0000$$

2. Expanding K_a and K_c by two round.

$$K_{a0} = 0000\ 0000\ 0000\ 0000$$
$$K_{a1} = 1111\ 1111\ 1111\ 1111$$
$$K_{a2} = 1010\ 0101\ 1010\ 0101$$

$$K_{c0} = 1111\ 0000\ 0000\ 0000$$
$$K_{c1} = 0000\ 0000\ 0000\ 0000$$
$$K_{c2} = 1100\ 1100\ 1100\ 1100$$

3. Perform XOR operation between K_a and $K_c$on each subkey for two rounds. So that the result of XOR K_a and K_c in each round is called key differences $\Delta K'$

$$\Delta K' = 1111\ 0000\ 0000\ 0000$$
$$\Delta K'_0 = 1111\ 0000\ 0000\ 0000$$
$$\Delta K'_1 = 1111\ 1111\ 1111\ 1111$$
$$\Delta K'_2 = 0110\ 1001\ 0110\ 1001$$

Steps to find key differences *key differences* K_d is as follows:

1. Perform an XOR operation between K_c and the 16-bit characteristic ΔK^*, resulting in the candidate key K_d.

$$K_c = 1111\ 0000\ 0000\ 0000$$
$$\Delta K^* = 1111\ 0000\ 1111\ 0000$$
$$K_d = 0000\ 0000\ 1111\ 0000$$

2. Expanding K_c and K_d by two round.

$$K_{c0} = 1111\ 0000\ 0000\ 0000$$
$$K_{c1} = 0000\ 0000\ 0000\ 0000$$
$$K_{c2} = 1100\ 1100\ 1100\ 1100$$

$$K_{d0} = 0000\ 0000\ 1111\ 0000$$
$$K_{d1} = 1111\ 1111\ 0000\ 0000$$
$$K_{d2} = 0011\ 1100\ 1100\ 1100$$

3. Perform XOR operation between K_c and K_d on each subkey for two rounds. So that the result of XOR K_c and K_d in each round is called key differences ΔK^*

$$\Delta K^* = 1111\ 0000\ 1111\ 0000$$
$$\Delta K^*_0 = 1111\ 0000\ 1111\ 0000$$
$$\Delta K^*_1 = 1111\ 1111\ 0000\ 0000$$
$$\Delta K^*_2 = 1111\ 0000\ 0000\ 0000$$

Step four, searching the related-key boomerang distinguisher on Mini-AES.

1. First generate 2^8 chosen plaintext P_0, then performs the XOR process P_0 with difference input α, to produce in 2^8 chosen plaintext P_0'. The difference input α is defined as 1111 0000 1111 0000 so that the plain text P_0 and P_0' will have the difference nibble values on byte position 0 and 2 as can be seen below:

$$\begin{aligned} P_0 &= 0000\ 0000\ 0000\ 0000 \\ &= 0000\ 0000\ 0001\ 0000 \\ &\vdots \\ &= 1111\ 0000\ 1111\ 0000 \end{aligned}$$

$$\begin{aligned} P_0' &= 1111\ 0000\ 1111\ 0000 \\ &= 1111\ 0000\ 1110\ 0000 \\ &\vdots \\ &= 0000\ 0000\ 0000\ 0000 \end{aligned}$$

2. The plain text P_0 is encrypted one round using the key K_a, The result will be notated as A_1 such that $A_1 = E_{K_a}(P_0)$. On the other hand, the P_0' is encrypted using one round using the key K_b such that $A_1' = E_{K_b}(P_0')$. The result of A_1 and A_1' are:

$$\begin{aligned} A_1 &= 0001\ 0001\ 0001\ 0001 \\ &= 0001\ 0001\ 1100\ 0110 \\ &\vdots \\ &= 1001\ 0000\ 1001\ 0000 \end{aligned}$$

$$\begin{aligned} A_1' &= 1110\ 1110\ 0001\ 0001 \\ &= 1110\ 1110\ 1100\ 0110 \\ &\vdots \\ &= 0110\ 1111\ 1001\ 0000 \end{aligned}$$

3. The plain text P_0 is encrypted two round using the key K_a, The result will be notated as C_2 such that $C_2 = E_{K_a}(P_0)$. On the other hand, the P_0' is encrypted using two round using the key K_b such that $C_2' = E_{K_b}(P_0')$. The result of C_2 and C_2' are:

$$\begin{aligned} C_2 &= 1110\ 0001\ 1110\ 0001 \\ &= 1110\ 1110\ 1111\ 0001 \\ &\vdots \\ &= 0000\ 1011\ 0000\ 1011 \end{aligned}$$

$$\begin{aligned} C_2' &= 0101\ 0001\ 1110\ 0101 \\ &= 0101\ 1110\ 1111\ 0101 \\ &\vdots \\ &= 1110\ 1011\ 0000\ 0010 \end{aligned}$$

4. The XOR operation between C_3 and C'_3 with $\delta = 1111\ 0000\ 0000\ 0000$ is performed, resulting in the values D_3 and D'_3.

$$\begin{aligned} D_2 &= 0001\ 0001\ 1110\ 0001 \\ &= 0001\ 1110\ 1111\ 0001 \\ &\vdots \\ &= 1111\ 1011\ 0000\ 1011 \end{aligned}$$

$$\begin{aligned} D'_2 &= 1010\ 0001\ 1110\ 0101 \\ &= 1010\ 1110\ 1111\ 0101 \\ &\vdots \\ &= 0001\ 1011\ 0000\ 0010 \end{aligned}$$

5. D_2 is decrypted one round using K_c, $B_2 = E'_{K_c}(D2)$, and also the D'_2 is decrypted one round using K_d, $B'_2 = E'_{K_d}(D'2)$. For knowing the value of β, an XOR operation is performed between B_2 with B'_2:

$$\begin{aligned} B_0 &= 0110\ 1101\ 1111\ 0101 \\ &= 0110\ 0111\ 0111\ 0000 \\ &\vdots \\ &= 0011\ 0001\ 1001\ 1100 \end{aligned}$$

$$\begin{aligned} B'_0 &= 1100\ 0010\ 1111\ 0101 \\ &= 1100\ 1011\ 0111\ 0000 \\ &\vdots \\ &= 1001\ 1110\ 1001\ 1100 \end{aligned}$$

6. D_2 is decrypted two round using K_c, $O_0 = E'_{K_c}(D2)$, and also the D'_2 is decrypted two round using K_d, $O'_0 = E'_{K_d}(D'2)$. To knowing the value of α, an XOR operation is performed between O_0 with O'_0:

$$\begin{aligned} O_0 &= 1011\ 0000\ 0111\ 0100 \\ &= 1011\ 0101\ 1000\ 0100 \\ &\vdots \\ &= 0101\ 1111\ 1000\ 0011 \end{aligned}$$

$$\begin{aligned} O'_0 &= 0100\ 0000\ 1000\ 0100 \\ &= 0100\ 0101\ 0111\ 0100 \\ &\vdots \\ &= 1000\ 1111\ 0111\ 1111 \end{aligned}$$

7. If $P_0 \oplus P'_0 = \alpha = O_0 \oplus O'_0$, $A_1 \oplus A'_1 = \beta = B_1 \oplus B'_1$, $A_1 \oplus B_1 = \gamma = A'_1 \oplus B'_1$, and $C_2 \oplus D_2 = \delta = C'_2 \oplus D'_2$, then it can be categorized as correct related-key boomerang quartet.

Table 2: The correct quartets on Related-Key Boomerang Attack

No	P_0	P'_0	O_0	O'_0
1.	0000000000000000	1111000011110000	1011000001110100	0100000010000100
2.	0000000000010000	1111000011100000	1011010110000100	0100010101110100
...	...	...	...	...
16.	0000000011110000	1111000000000000	1011111110000100	0100111101110100
17.	1010000000000000	0101000011110000	1101000001111101	0010000010001101
18.	1010000000010000	0101000011100000	1101010110001101	0010010101111101
...	...	...	...	...
32.	1010000011110000	0101000000000000	1101111110001101	0010111101111101
33.	1110000000000000	0001000011110000	1001000001111101	0110000010001101
34.	1110000000010000	0001000011100000	1001010110001101	0110010101111101
...	...	...	...	...
48.	1110000011110000	0001000000000000	1001111110001101	0110111101111101

In this study, pairs of $P_0, P'_0, O_0, \text{dan} O'_0$ were found in Table 2 which is the correct related-key boomerang quartet because it satisfies the equation $P_0 \oplus P'_0 = \alpha = O_0 \oplus O'_0$, $A_1 \oplus A'_1 = \beta = B_1 \oplus B'_1$, $A_1 \oplus B_1 = \gamma = A'_1 \oplus B'_1$, and $C_2 \oplus D_2 = \delta = C'_2 \oplus D'_2$. This study, for instance using P_0 0000000000000000 and P'_0 1111000011110000, the α value obtained is 1111000011110000. Then, calculating the value A_1 0001000100010001 and A'_1 1110111000010001 with the β value 1111111100000000. Next, calculating the value C_2 1110000111100001 and C'_2 0101000111100101, with the δ value 1111000000000000, calculating the value D_2 0001000111100001 and D'_2 1010000111100101. After that, calculating the value B_1 0010001001000010 and B'_1 1101110101000010 yields the γ value 0011001101010011. Finally, calculating the value O_0 1011000001110100 and O'_0 0100000010000100, and ensuring that the α value equals 1111000011110000. Using 2^8 plaintext pairs, we had succesfully obtained 48 correct quartet pairs with probability $P_{r_c} = p \cdot \tilde{p} \cdot \tilde{q}^2 = 2^{-4} \cdot 2^{-4} \cdot 2^{-4} = 2^{12}$. Because every chosen plaintext is generated arbitrarily but has differences in byte positions 0 and 2, then it will produce a correct related key boomerang quartet. The greater the probability, the easier it is to attack an algorithm. The correct related-key boomerang distinguishers found can be used to recover the key using the steps that are already mentioned in the section 2.

4 Conclusion

In this research as mentioned above, the related-key boomerang attack on mini-AES is conducted under scenario of differential (α, β) = (1111 0000 1111 0000, 1111 1111 0000 0000) and (δ, γ) = (1111 0000 0000 0000, 0011 0011 1000 0001) with probability $p = 2^4$ and $q = 2^4$ respectively. Using 2^8 plaintext pairs, the attack succesfully delivered 48 correct quarted pairs with the total probability of 2^{12}. It is suggested that for validation purposes, key recovery should be performed using one of the distinguishers that has already been identified. This attack reveals a security hole in Mini-AES and provides important insights into vulnerabilities in similar cryptographic systems. Through in-depth analysis, this attack provides a better understanding of the complexity of Mini-AES security and its relevance in the context of related-key attacks. As a result, a valuable contribution is made by this research in strengthening the understanding of the security of cryptographic systems and emphasizing the importance of protection against key-related vulnerabilities in encryption algorithms.

References

[1] Phan, R. C. W. (2002). Mini advanced encryption standard (mini-AES): a testbed for cryptanalysis students. Cryptologia, 26(4), 283-306.

[2] Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

[3] Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology 7(4), 229–246 (1994)

[4] Biryukov, A.: The Boomerang Attack on 5 and 6-Round Reduced AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 11–15. Springer, Heidelberg (2005)

[5] Gorski, M., & Lucks, S. (2008). New related-key boomerang attacks on AES. In Progress in Cryptology-INDOCRYPT 2008: 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings 9 (pp. 266-278). Springer Berlin Heidelberg.

[6] von zur Gathen, J., & von zur Gathen, J. (2015). Chapter 6 Differential and linear cryptanalysis. CryptoSchool, 263-300.

[7] Biham, Eli, and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." Journal of CRYPTOLOGY 4 (1991): 3-72.

[8] Biham, E., Dunkelman, O., & Keller, N. (2005). Related-key boomerang and rectangle attacks. In Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24 (pp. 507-525). Springer Berlin Heidelberg.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 110–117.

Neural Network–based Cryptanalysis of PRESENT and D-PRESENT Block Ciphers

Ronniel D. Labio[1] and Enrique Festijo[1,2]

Technological Institute of the Philippines, Quezon City, Philippines
rd.labio@ndmc.edu.ph,enrique.festijo@tip.edu.ph

Abstract. This paper evaluates the security of the PRESENT and D-PRESENT block ciphers using Multilayer Perceptrons (MLPs) Neural Networks for known plaintext attacks. We analyzed three neural network architectures—64-128-128-64, 128-256-256-128, and 256-512-512-256—finding that the 256-512-512-256 configuration provided the best performance for both ciphers. The optimal Mean Squared Error (MSE) was 0.238141032 and 0.238146656, respectively. The Test Error was 0.48802551 for PRESENT and 0.48810985 for D-PRESENT, indicating that both ciphers exhibit strong resistance to cryptanalysis. This study highlights the use of neural networks in evaluating cryptographic algorithms and confirms the robustness of both PRESENT and D-PRESENT against advanced analytical techniques.

Keywords: Cryptanalysis · PRESENT · D-PRESENT · Neural Networks · Mean Squared Error · Test Error

1 Introduction

Cryptanalysis involves examining and breaking cryptographic systems to identify weaknesses or vulnerabilities. The process of identifying weaknesses in cryptographic algorithms and utilizing them to decipher ciphertext without the secret key has evolved from the inception of Differential cryptanalysis, the earliest general cryptanalytic method, to various techniques such as linear cryptanalysis and integral analysis. Multiple research endeavors [1,2,3,4,5,6] have provided evidence of successful cryptanalysis on block ciphers. Nonetheless, as indicated by [7], these traditional cryptanalytic approaches may have practical limitations or constraints on their applicability. Typically, conventional cryptanalytic methodologies demand considerable time, access to known plaintexts, and memory resources. Moreover, while traditional cryptanalysis is generally conducted without restrictions on the keyspace, recent block ciphers have only been vulnerable to attacks on reduced-round variants [7].

Recent advancements in machine learning (ML), particularly Neural Networks, have introduced new paradigms in cryptanalysis. Neural Networks have proven to be highly effective in enhancing cryptanalysis performance, as evidenced by several studies. [8] showed that neural networks could identify patterns and potential weaknesses in the key scheduling algorithms of block ciphers, providing insights into their security. In [9], a deep learning-based model was developed to predict block cipher keys from known plaintext-ciphertext pairs, with successful applications to lightweight block ciphers such as Simon, Speck, and simplified DES. It was found that deep learning-based cryptanalysis could recover key bits when the key space was limited to 64 ASCII characters. Likewise, [10] utilized neural networks to effectively perform differential cryptanalysis on the Speck cipher, achieving remarkable success rates compared to traditional methods. Similarly, [11] explored the application of deep learning techniques to break various modes of operation of

the Advanced Encryption Standard (AES). The research demonstrated that deep learning models could effectively identify weaknesses and patterns in the encrypted data, thus allowing the decryption of ciphertext without knowing the secret key.
Furthermore, [12] explores the use of neural networks in cryptography, specifically focusing on analyzing block ciphers using statistical methods. The Inception V3 neural network model, typically used for image identification, was repurposed to differentiate ciphertexts of block ciphers. Results indicated that neural networks can effectively address challenges in analyzing statistical properties. In [13], two experiments aimed to recover the master key bit by bit of PRESENT. Both techniques, employing a 3-depth Fully Connected Neural Network and a residual neural network, demonstrated high accuracy in predicting the security key of PRESENT. However, [14] examined the security of the key scheduling algorithm (KSA) in the 80-bit key variant of PRESENT by employing deep learning techniques to attempt the extraction of the main key register from the final round key register. The results indicated that the KSA of PRESENT offers strong security. Similarly, [15] examined the KSA of AES and PRESENT using a deep learning model to identify patterns and weaknesses in these block ciphers. The results indicated that only half of the final round key bits could be accurately predicted, suggesting that these KSAs are effective at resisting certain types of deep learning analysis.
D-PRESENT is a modified version of PRESENT, designed to enhance security by incorporating a dynamic, key-dependent S-box. While the original PRESENT cipher is well-known for its simplicity and efficiency, utilizing a static S-box across all encryption rounds, D-PRESENT introduces variability by altering the S-box based on the round key for every encryption round. This dynamic modification makes the S-box in D-PRESENT dependent on the specific key used, adding an extra layer of complexity and making cryptanalysis more challenging. Additionally, while D-PRESENT retains the same bit permutation layer (pLayer) as PRESENT, the key-dependent S-box distinguishes it by introducing non-linearity that adapts with each encryption, thereby enhancing resistance to certain types of attacks.
Building on the strengths of PRESENT, D-PRESENT has been introduced as an improved variant, offering enhancements in the avalanche effect, execution time, and throughput [16]. The avalanche effect guarantees that even a minor alteration in the plaintext or key results in a significantly different ciphertext, thereby strengthening security. Improved execution time and throughput make D-PRESENT more efficient and suitable for resource-constrained environments. These pioneering efforts illustrate the potential of neural network-based approaches to enhance the cryptanalysis of lightweight ciphers. While much of the research focuses on recovering secret keys to decrypt ciphertext generated by block ciphers like PRESENT, this study aims to decrypt ciphertext into plaintext using known ciphertext-plaintext datasets without the need to recover the secret key. It employs Multilayer Perceptron (MLPs) or feedforward neural networks to evaluate the security robustness of PRESENT and D-PRESENT by assessing their performance based on metrics such as Mean Squared Error (MSE) and Test Error (TE).
The results of this study offer valuable insights into the robustness of the PRESENT and D-PRESENT block ciphers when subjected to neural network-based cryptanalysis. The performance of both ciphers to known plaintext attacks suggests their security strength even when modern machine learning techniques are employed. The result is particularly relevant for applications in resource-constrained environments where lightweight ciphers are preferred, such as in IoT devices and embedded systems. Moreover, the use of neural networks in assessing cipher security can serve as an additional layer of analysis, complementing traditional methods like differential and linear cryptanalysis. As neural networks continue to evolve, understanding their strengths and limitations in cryptographic analysis will be crucial for designing future ciphers that are secure against a broader range of attack vectors.

Table 1: Experimental Equipment

OS Version	Windows 10 Pro 22H2
Processor	11th Gen Intel® Core™ i7-11700 @2.5 Ghz
RAM	8.00 Gb
Graphics Card	Intel® UHD Graphics 750

The rest of this paper is structured as follows: Section 2 outlines the methodology used in this research. Section 3 presents the experimental results. Section 4 presents the analysis. Finally, Section 5 concludes the paper with insights and future research directions.

2 Methods of Experimentation

2.1 Experimental Environment

The experiments were conducted using MATLAB R2023b. Table 1 displays the equipment utilized for the experiments. The choice of hardware was guided by practical constraints in research settings. The results obtained are still relevant for understanding the baseline performance and potential of the approach.

2.2 Datasets

The experiment utilized 5,000 pairs of known 64-bit ciphertext and plaintext datasets for training the neural network, along with 100 pairs for testing the model's actual performance. This sample size provides a balance between demonstrating the technique's capabilities and managing computational resources. The goal of the research to demonstrate the feasibility of an attack can still be achieved.

2.3 Parameters

The parameters of the neural network were set as follows:

2.3.1 Hidden Layers

The neural network was trained with three different hidden layer configurations: 64-128-128-256, 128-256-256-128, and 256-512-512-256, as recommended in [16]. Figure 1 shows the sample network setting of the experimentation.

2.3.2 Training Function

The training function employed was the quantized conjugate gradient method, 'trainscg', suitable for large networks [17].

2.3.3 Error Function

The error function utilized for training was the mean squared error (MSE), which is commonly applied in regression tasks, such as restoring plaintext from known ciphertext.

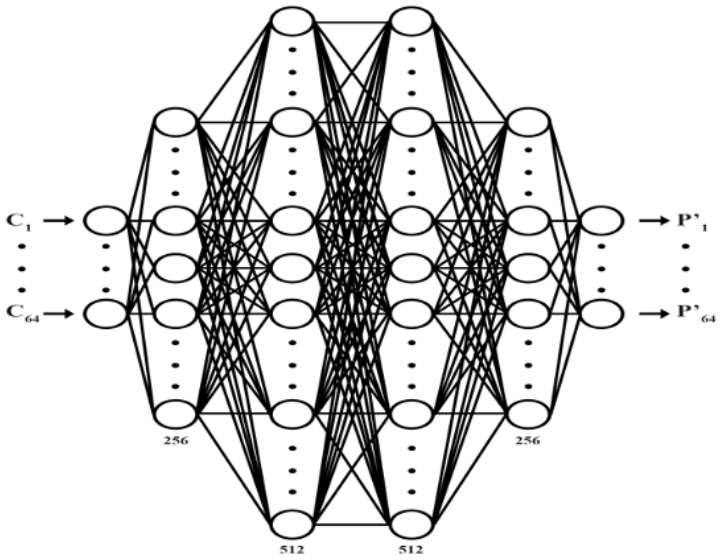


Figure 1: Experimental training network.

2.4 Training Stop Conditions

The maximum number of training cycles was set to 500, with three stop conditions: (1) reaching 500 training cycles, (2) achieving an acceptable mean square error limit of 0.05, and (3) experiencing continuous validation failures, with a maximum limit of 20 failures.

2.5 Training Phase

The collected dataset was divided into training samples (70%), validation samples (15%), and testing samples (15%). Training samples were used to adjust the connection weights of the network based on error. The same dataset was used to train the network with different hidden layer configurations, and multiple training trials were conducted to identify the best performance.

2.6 Test Error Phase

After determining the best performance of the network with the three different layer settings, the lowest MSE was selected to assess the network's capability to successfully attack the block cipher. The ciphertext different from the training set was input into the neural network, generating the corresponding result P'_{test}. Results with values less than 0.5 were interpreted as bit 0, otherwise as bit 1. The P'_{test} and the plaintext P_{test} sets were XORed bit by bit, and the error percentage was calculated as the test error, expressed by the following formula:

$$test\_error = \frac{\sum_{i=1}^{m_{test}} \sum_{j=1}^{n_{test}} p'_{test}(i,j) \oplus p_{test}(i,j)}{m_{test} \times n_{test}} \tag{1}$$

Table 2: Performance result for PRESENT.

Hidden Layers	Performance (MSE)	Epoch	Elapsed Time
64-128-128-64	0.240078647	133	10 s
128-256-256-128	0.238811549	173	25 s
256-512-512-256	0.238141032	100	40 s

Table 3: Performance result for D-PRESENT.

Hidden Layers	Performance (MSE)	Epoch	Elapsed Time
64-128-128-64	0.24021601	136	16 s
128-256-256-128	0.238822611	182	28 s
256-512-512-256	0.238146656	98	43 s

where

m_{test} is the number of bits per block in the test set,
n_{test} is the number of blocks in the test set,
P'_{test} (i,j) is the *jth* bit in the *ith* block of the output during the test, and
P_{test} (i,j) is the *jth* bit in the *ith* block of the plaintext used in the test.

3 Results

After carrying out experimentation on PRESENT and D-PRESENT block ciphers, the following results were recorded.

3.1 Mean Squared Error (MSE)

The mean squared error (MSE) is a measure used to evaluate the accuracy of a predictive model. It is calculated as the average of the squared differences between the actual values and the predicted values. There is no correct value for MSE, however, the lower the value the better.

Table 2 presents the MSE values for the PRESENT block cipher under a known plaintext attack using a neural network. The results show that the network achieves its best performance with a configuration of 256-512-512-256 hidden layers, recording an MSE of 0.238141032. As observed, increasing the number of neurons in each layer generally leads to a reduction in MSE.

Table 3 displays the MSE values for the D-PRESENT block cipher. Similar to PRESENT, the optimal performance is achieved with the 256-512-512-256 hidden layers configuration, with an MSE of 0.238146656. Again, a trend is observed where increasing the number of neurons in each layer results in a lower MSE.

3.2 Test Error

Test error quantifies the real-world performance of a neural network when it encounters new data that was not part of its training set. It assesses the discrepancy between the predicted output (P'_{test}) and the actual output (P_{test}) when the network is presented with unseen ciphertext. Essentially, test error measures the number of errors that the network fails to predict accurately during inference.

Table 4: Test Error Result.

Hidden Layers	PRESENT	D-PRESENT
64-128-128-64	0.49348485	0.48516414
128-256-256-128	0.49195076	0.48869318
256-512-512-256	0.48802551	0.48810985

Table 4 shows the test error results for both block ciphers. For both PRESENT and D-PRESENT, the lowest test error is observed with the 256-512-512-256 hidden layers configuration, recording values of 0.4880 and 0.4881, respectively. As with MSE, an increase in the number of neurons tends to reduce the test error for both algorithms.

4 Discussion

The network's optimal performance for both, PRESENT and D-PRESENT, was achieved using a hidden layer configuration of 256-512-512-256, resulting in an MSE of 0.238141032 after 100 epochs and 40 seconds of elapsed time for PRESENT and an MSE of 0.238146656 after 98 epochs and 43 seconds of elapsed time for D-PRESENT. A clear trend was observed for both block ciphers indicating that an increase in the number of neurons per layer correlates with a decrease in MSE.

Thus, achieving the best network performance necessitates additional elapsed time. In addition, the lowest test error was observed with the 256-512-512-256 hidden layer configuration, yielding a test error of 0.48810985 for PRESENT and 0.48810985 for D-PRESENT. It was further noted that as the number of neurons in each layer increased, the test error correspondingly decreased.

The results from our experimentation indicate that PRESENT and D-PRESENT block ciphers exhibit robust security against known plaintext attacks when assessed using neural networks, specifically Multilayer Perceptrons. This robustness is evidenced by the relatively low Mean Squared Error (MSE) and Test Error (TE) across different neural network configurations. These findings highlight the potential of PRESENT and D-PRESENT as a secure option in applications requiring lightweight cryptography, particularly where conventional cryptanalytic attacks might be mitigated but machine learning-based attacks pose a new threat.

The practical implications of our findings are twofold. Firstly, they affirm the security of both PRESENT and D-PRESENT in protecting sensitive information within constrained environments, such as IoT devices. Secondly, the use of neural networks in cryptanalysis provides a new dimension of evaluation, helping cryptographers understand and reinforce the strengths and weaknesses of their algorithms against both traditional and modern attack vectors.

5 Conclusion

This research aimed to assess the security resilience of PRESENT and D-PRESENT against known plaintext attacks using a Multilayer Perceptron neural network. The results indicate that while there is a slight decrease in mean squared error and test error with an increase in the number of neurons in each layer, this decrease is marginal and does not hold significant practical implications. Moreover, as the number of neurons increases, there is a notable increase in the time required to train the network. Consequently, developing a more accurate predictive neural network model necessitates longer training durations and higher computing resources in terms of speed and memory.

Nevertheless, the findings affirm that both block ciphers exhibit robust security against known plaintext attacks conducted using neural networks. This study supports previous research indicating that the PRESENT block cipher stands up well against deep learning-based analyses [14,15].

Recommendation

While the current study used Multilayer Perceptrons (MLPs), experimenting with other architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) could provide deeper insights into the cryptanalysis process. These architectures might capture more complex patterns in the data.

Moreover, an increase in the number of datasets and upgrading of computer resources, like RAM and Processor, may also provide insightful results.

References

[1] Liu, Y., Li, Y., Chen, H., & Wang, M. (2023). Full-round impossible differential attack on shadow block cipher. Cybersecurity, 6(1), 52.
[2] Chan, Y. Y., Khor, C. Y., Khoo, B. T., Teh, J. S., Teng, W. J., & Jamil, N. (2023). On the resistance of new lightweight block ciphers against differential cryptanalysis. Heliyon, 9(4).
[3] Fayyaz, S. (2024). Trend Analysis and Statistical Cryptoanalysis of Light Weight Block Ciphers (No. 13715). EasyChair.
[4] Zhu, S., Wang, G., He, Y., & Qian, H. (2020). Integral attacks on some lightweight block ciphers. KSII Transactions on Internet and Information Systems (TIIS), 14(11), 4502-4521.
[5] Teh, J. S., Tham, L. J., Jamil, N., & Yap, W. S. (2022). New differential cryptanalysis results for the lightweight block cipher BORON. Journal of Information Security and Applications, 66, 103129.
[6] Yeo, S. L., Le, D. P., & Khoo, K. (2021). Improved algebraic attacks on lightweight block ciphers. Journal of cryptographic Engineering, 11, 1-19.
[7] Anees, A., Hussain, I., Khokhar, U. M., Ahmed, F., & Shaukat, S. (2022). Machine learning and applied cryptography. Security and Communication Networks, 2022, 1-3.
[8] Lee, T. R., Teh, J. S., Yan, J. L. S., Jamil, N., & Yeoh, W. Z. (2020). A machine learning approach to predicting block cipher security. In Universiti Putra Malaysia (pp. 122-132).
[9] [So, J. (2020). Deep learning-based cryptanalysis of lightweight block ciphers. Security and Communication Networks, 2020, 1-11.
[10] Gohr, A. (2019). Improving attacks on round-reduced speck32/64 using deep learning. In Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39 (pp. 150-179). Springer International Publishing.
[11] Perusheska, M. G., Trpceska, H. M., & Dimitrova, V. (2022, March). Deep learning-based cryptanalysis of different AES modes of operation. In Future of Information and Conference (pp. 675-693). Cham: Springer International Publishing.
[12] Perov, A. (2019, October). Using machine learning technologies for carrying out statistical analysis of block ciphers. In 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) (pp. 0853-0856). IEEE.
[13] Duan, M., Zhou, R., Fu, C., Guo, S., & Wu, Q. (2021, December). Vulnerability Testing on the Key Scheduling Algorithm of PRESENT Using Deep Learning. In International Conference on Security and Privacy in New Computing Environments (pp. 307-318).

Cham: Springer International Publishing.
[14] Pareek, M., Mishra, G., & Kohli, V. (2020). Deep learning based analysis of key scheduling algorithm of PRESENT cipher. Cryptology ePrint Archive.
15] Patel, N. K., & Lamkuche, H. S. (2024). Deep Learning Based Analysis of Key Scheduling Algorithm of Advanced Ciphers. Cryptology ePrint Archive.
[16] Labio, R. D., & Festijo, E. D. (2020, October). D-present: a lightweight block cipher with dynamic key-dependent substitution boxes. In 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 27-32). IEEE.
[17] Alani, M. M. (2019, January). Applications of machine learning in cryptography: a survey. In Proceedings of the 3rd International Conference on cryptography, security and privacy (pp. 23-27).],
[18] Hu, X., & Zhao, Y. (2018). Research on plaintext restoration of AES based on neural network. Security and Communication Networks, 2018, 1-9.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 118–139.

Durian: A General-Purpose Block Cipher

Muhammad Reza Z'aba and Mohd Aminudin Mohd Khalid

Cryptography Lab, MIMOS Berhad, MRANTI Technology Park, 57000 Kuala Lumpur
reza.zaba@mimos.my aminudin.khalid@mimos.my

Abstract. This article presents the block cipher Durian, a general-purpose block cipher suitable for implementation across a wide range of platforms. Durian belongs to the class of reflection ciphers, characterised by identical encryption and decryption algorithms except for the processing of the round subkeys. Unlike traditional Feistel-based ciphers, reflection ciphers feature an involutive component in the middle of the cipher. In addition to describing the specification of Durian, this article also provides an in-depth analysis of its security and performance, along with the rationale behind its design choices.

Keywords: Block cipher · Reflection ciphers · Generalized Feistel Network · Symmetric cryptography

1 Introduction

Block ciphers are the workhorses of cryptography. They can be utilised to build other cryptographic primitives. If used in modes such as Counter (CTR), it can be used to build stream ciphers. In the Miyaguchi-Preneel construction, it can be realised as hash functions. Plug the block cipher into the CBC-MAC construction, and we will get a message authenticated code (MAC). Similarly, using the block cipher in Galois counter mode (CGM), we will get an authenticated encryption scheme. As the Advanced Encryption Standard (AES) [21] already is a fairly good block cipher, exploring other alternatives is still viable, as these may offer unique advantages compared to the AES.

Block ciphers like AES do not have identical encryption and decryption algorithms. For example, the matrix used in the `MixColumns` component for encryption differs from the matrix used in its inverse for decryption. Consequently, when AES is used in modes such as cipher block chaining (CBC) and OCB3 [31], both AES encryption and decryption algorithms are respectively required to perform mode-level encryption and decryption. In contrast, reflection block ciphers [16], with identical encryption and decryption algorithms, can use the same algorithm for both mode-level encryption and decryption. This approach saves area in resource-constrained environments and simplifies the construction logic by eliminating the need to select between block cipher encryption and decryption algorithms.

Reflection block ciphers are a subset of involutional block ciphers, like Noekeon [20], Anubis [2], and Khazad [3], which use involutive components (e.g. S-box) in their construction. The Data Encryption Standard (DES) [35] is also involutive due to its use of the standard Feistel network. The lightweight block cipher PRINCE [15] and PRINCE v2 [17] achieve the involutive property by incorporating an involutive component in the middle and ensuring the second half of the cipher is the inverse of the first half. These ciphers, known as *reflection ciphers* [16], have encryption functions identical to decryption functions. Beyne and Chen further provided a provable security treatment of these ciphers, reinforcing the robustness of this construction [4].

This article introduces the Durian block cipher, which is a type of reflection block ciphers by PRINCE. It accepts a 128-bit plaintext block and master key lengths of 128,

192 or 256 bits. These variants are denoted as Durian-128, Durian-192 and Durian-256, respectively. The secret key is used as input to the key scheduling algorithm to produce a set of whitening and round subkeys. These key materials are used by the encryption and decryption algorithms to process the input block. The number of rounds is 16 for Durian-128, 20 for Durian-192 and 24 for Durian-256. The encryption and decryption algorithms employ a slightly modified Type-2 generalized Feistel network (GFN). With the introduction of the I function, the modification allows the use of the same algorithm to perform both encryption and decryption in the 4-branch setting. Traditional 4-branch Type-2 GFN uses two distinct algorithms.

This article is organized as follows. Section 2 describes how Durian performs encryption and decryption while Section 3 explains how it processes the secret key. Section 4 contains a security analysis of the cipher. The design rationale is presented in Section 5. An efficient Durian implementation strategy is described in Section 6 while the performance figures for software and hardware are provided in Section 7.

2 Processing the Plaintext Block

This section describes how the 128-bit input block is processed by Durian during encryption and decryption. The plaintext block $\breve{p} = \breve{p}_0\|\breve{p}_1\|\breve{p}_2\|\breve{p}_3$ and the ciphertext block $\breve{c} = \breve{c}_0\|\breve{c}_1\|\breve{c}_2\|\breve{c}_3$ consist of the concatenation of four 32-bit words $\breve{p}_i$ and $\breve{c}_i$, respectively. The 32-bit whitening keys wk_i and the round subkeys rk_i are produced by the key scheduling algorithm described in Section 3. Let $\breve{R} \in \{16, 20, 24\}$ denote the number of rounds in Durian and $\breve{r} = \breve{R}/2$ denote the number of half rounds.

2.1 Encryption

Given the plaintext block $\breve{p} = \breve{p}_0\|\breve{p}_1\|\breve{p}_2\|\breve{p}_3$, whitening keys wk_i and round subkeys rk_j as inputs, encryption proceeds as follows:

1. Set the input block and key materials as follows:

$$\begin{aligned} \breve{t}_i &= \breve{p}_i && \text{for } i = 0, 1, \ldots, 3, \\ \breve{wk}_i &= wk_i && \text{for } i = 0, 1, \ldots, 7, \\ \breve{rk}_i &= rk_i && \text{for } i = 0, 1, \ldots, 2\breve{R} - 1. \end{aligned}$$

2. XOR the input block with the pre-whitening key $(\breve{wk}_0, \breve{wk}_1, \breve{wk}_2, \breve{wk}_3)$:

$$\begin{aligned} \breve{x}_0 &= \breve{t}_0 \oplus \breve{wk}_0, & \breve{x}_1 &= \breve{t}_1 \oplus \breve{wk}_1, \\ \breve{x}_2 &= \breve{t}_2 \oplus \breve{wk}_2, & \breve{x}_3 &= \breve{t}_3 \oplus \breve{wk}_3. \end{aligned}$$

3. Apply the following equations for $i = 1, 2, \ldots, \breve{r}$:

$$\begin{aligned} \breve{x}_{4i} &= F(\breve{rk}_{2i-2}, \breve{x}_{4i-4}) \oplus \breve{x}_{4i-3}, & \breve{x}_{4i+1} &= \breve{x}_{4i-2}, \\ \breve{x}_{4i+2} &= F(\breve{rk}_{2i-1}, \breve{x}_{4i-2}) \oplus \breve{x}_{4i-1}, & \breve{x}_{4i+3} &= \breve{x}_{4i-4} \end{aligned}$$

   where F is a bijective function defined in Section 2.3.1.

4. Apply the I function to the current value of the state (note the order of the input words)

$$\breve{y}_{4\breve{r}}\|\breve{y}_{4\breve{r}+1}\|\breve{y}_{4\breve{r}+2}\|\breve{y}_{4\breve{r}+3} = I(\breve{x}_{4\breve{r}+3}\|\breve{x}_{4\breve{r}}\|\breve{x}_{4\breve{r}+1}\|\breve{x}_{4\breve{r}+2})$$

   where I is an involutive function defined in Section 2.3.4.

5. Apply the following equations for $i = \breve{r}+1, \breve{r}+2, \ldots, \breve{R}$:

$$\breve{y}_{4i} = F(\breve{rk}_{2i-1}, \breve{y}_{4i-2}) \oplus \breve{y}_{4i-1}, \qquad \breve{y}_{4i+1} = \breve{y}_{4i-4},$$
$$\breve{y}_{4i+2} = F(\breve{rk}_{2i-2}, \breve{y}_{4i-4}) \oplus \breve{y}_{4i-3}, \qquad \breve{y}_{4i+3} = \breve{y}_{4i-2}.$$

6. XOR the state with the post-whitening keys $(\breve{wk}_4, \breve{wk}_5, \breve{wk}_6, \breve{wk}_7)$ and output:

$$\breve{t}_4 = \breve{y}_{4\breve{R}+1} \oplus \breve{wk}_4, \qquad \breve{t}_5 = \breve{y}_{4\breve{R}+2} \oplus \breve{wk}_5,$$
$$\breve{t}_6 = \breve{y}_{4\breve{R}+3} \oplus \breve{wk}_6, \qquad \breve{t}_7 = \breve{y}_{4\breve{R}} \oplus \breve{wk}_7.$$

7. Output the ciphertext block as

$$\breve{c}_0 = \breve{t}_4, \qquad \breve{c}_1 = \breve{t}_5, \qquad \breve{c}_2 = \breve{t}_6, \qquad \breve{c}_3 = \breve{t}_7.$$

The encryption for Durian is depicted in Figure 1(a) (note that the pre- and post-whitening keys are not shown). A high-level description of the cipher and its components are given in Listing 1 and Listing 2, respectively.

2.2 Decryption

Given the ciphertext block $\breve{c} = \breve{c}_0 \| \breve{c}_1 \| \breve{c}_2 \| \breve{c}_3$, whitening keys wk_i and round subkeys rk_j as inputs, decryption is identical to encryption except for the order of the round subkeys. To decrypt, the following values are used

$$\begin{aligned}
\breve{t}_i &= \breve{c}_i && \text{for } i = 0, 1, \ldots, 3,\\
\breve{wk}_i &= wk_{i+4} && \text{for } i = 0, 1, \ldots, 3,\\
\breve{wk}_i &= wk_{i-4} && \text{for } i = 4, 5, \ldots, 7,\\
\breve{rk}_i &= rk_{2\breve{R}-i-2} && \text{for } i = 0, 2, 4, \ldots, 2\breve{R}-2,\\
\breve{rk}_i &= rk_{2\breve{R}-i} && \text{for } i = 1, 3, 5, \ldots, 2\breve{R}-1.
\end{aligned}$$

Then, apply Steps 2 to 6 of the encryption procedure described in the previous section and output the plaintext block as

$$\breve{p}_0 = \breve{t}_4, \qquad \breve{p}_1 = \breve{t}_5, \qquad \breve{p}_2 = \breve{t}_6, \qquad \breve{p}_3 = \breve{t}_7.$$

The decryption for Durian is depicted in Figure 1(a) (note that the pre- and post-whitening keys are not shown). The order of the whitening and round subkeys for decryption have to be amended accordingly to fit the high-level description of Listing 1.

2.3 Components

This section explains the components used in the encryption and decryption algorithms of Durian.

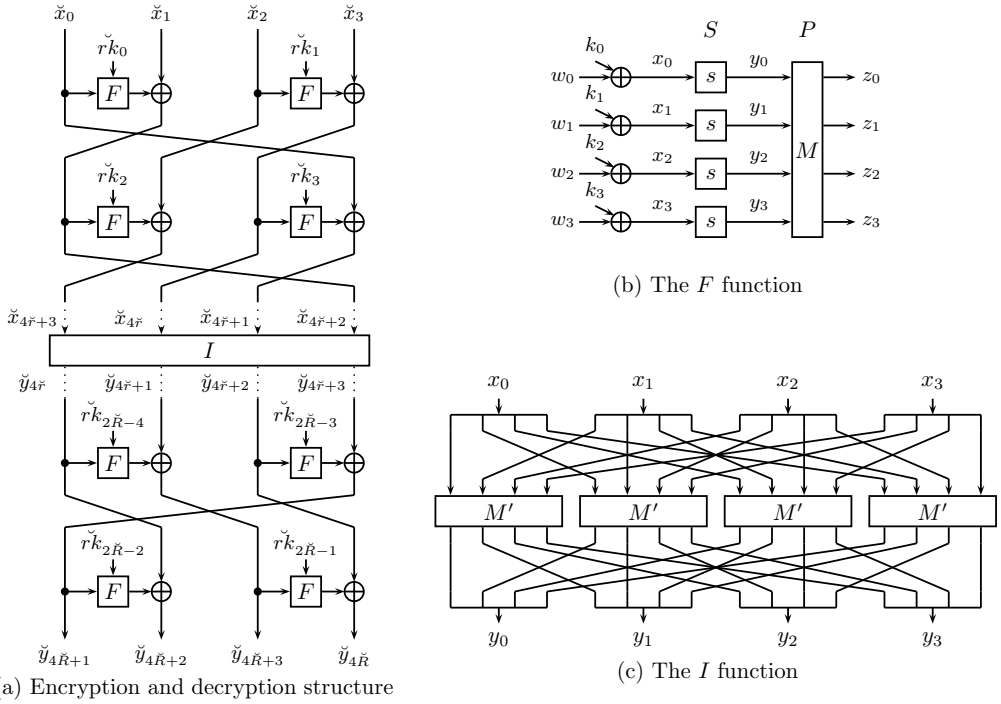
2.3.1 The F Function

The F function accepts two 32-bit inputs: a round subkey[1] k and a state word w. It produces a 32-bit output z and consists of a nonlinear transformation S and a linear transformation P as follows

$$z = F(k, w) = P(S(k \oplus w)).$$

The F function is depicted in Figure 1(b) where $k = k_0 \| k_1 \| k_2 \| k_3$ and $w = w_0 \| w_1 \| w_2 \| w_3$ are treated as the concatenation of four 8-bit words k_i and w_i, respectively.

[1] For simplicity, the notation k here is used in the generic sense to represent rk_i and $\breve{rk}_i$ stated in Section 2.1 and Figure 1(a).



(a) Encryption and decryption structure

(b) The F function

(c) The I function

Figure 1: Components of Durian

2.3.2 The S Function

The S function is a nonlinear transformation that accepts a 32-bit word $x = x_0\|x_1\|x_2\|x_3$ as input which consists of a sequence of four 8-bit words x_i. The application of S to x comprises the application of a single 8×8 S-box s to x_i four times in parallel to produce a 32-bit output y. The S function is denoted as

$$y = y_0\|y_1\|y_2\|y_3 = S(x) = s(x_0)\|s(x_1)\|s(x_2)\|s(x_3).$$

The mapping for the 8×8 S-box s is given in Table 1. The left-most column denotes the first four bits and the top-most row denotes the subsequent four bits of the 8-bit input of the S-box. For instance, for input `12` (in hexadecimal) to the S-box s, the output is $s(\texttt{12}) = \texttt{A4}$.

2.3.3 The P Function

The linear P function accepts a 32-bit word $y = y_0\|y_1\|y_2\|y_3$ as input which consists of four 8-bit words y_i. The P function comprises the application of a 4×4 Maximum Distance Separable (MDS) matrix M to y to produce a 32-bit output word $z = z_0\|z_1\|z_2\|z_3$. The MDS matrix M is:

$$M = \begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix}.$$

The application of P to y is denoted by

$$z = P(y)$$

Table 1: The S-box table of Durian

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	2D	1C	65	E8	2C	8F	8D	29	E0	F8	02	B0	A1	36	5C	8B
1	4A	7C	A4	14	A3	23	5F	37	41	80	AD	5E	59	DC	89	68
2	58	1A	F5	F9	FD	2F	A2	31	E2	BB	38	95	CD	00	9A	C6
3	0B	B3	DA	3C	0D	26	FA	DE	E5	99	83	B4	2A	A5	57	BF
4	D2	C2	4D	F3	DF	C1	93	54	18	AF	74	24	D5	2B	B2	C3
5	E9	85	0C	FB	42	70	4F	E1	9D	B8	D7	7F	97	3B	CE	E6
6	53	33	CB	96	AB	F0	C9	43	7B	E7	84	C4	07	06	20	86
7	F6	A7	03	56	4E	CF	D4	0F	C8	34	CA	35	0A	1D	87	19
8	6C	B1	45	A6	47	01	F7	46	17	04	D1	BE	67	10	9E	F4
9	EE	9C	AE	DD	32	E3	27	40	73	1E	FF	15	FC	ED	72	C7
A	BA	C0	66	6E	4C	3A	30	48	9F	6B	B9	BD	E4	F1	7D	90
B	88	EB	98	8E	D0	22	61	7E	EC	A9	D6	91	09	2E	62	8C
C	81	08	11	50	B5	B7	DB	7A	25	6A	63	8A	16	5B	A8	77
D	EA	FE	55	76	51	78	6D	71	1F	60	28	13	AC	6F	F2	5D
E	4B	3F	69	BC	94	5A	3D	A0	D3	B6	3E	12	44	1B	D8	92
F	21	75	0E	CC	82	79	39	49	64	EF	52	AA	C5	9B	D9	05

and can be written as

$$\begin{aligned} z_0 &= \mathsf{02} \cdot y_0 \oplus \mathsf{03} \cdot y_1 \oplus \mathsf{01} \cdot y_2 \oplus \mathsf{01} \cdot y_3, \\ z_1 &= \mathsf{01} \cdot y_0 \oplus \mathsf{02} \cdot y_1 \oplus \mathsf{03} \cdot y_2 \oplus \mathsf{01} \cdot y_3, \\ z_2 &= \mathsf{01} \cdot y_0 \oplus \mathsf{01} \cdot y_1 \oplus \mathsf{02} \cdot y_2 \oplus \mathsf{03} \cdot y_3, \\ z_3 &= \mathsf{03} \cdot y_0 \oplus \mathsf{01} \cdot y_1 \oplus \mathsf{01} \cdot y_2 \oplus \mathsf{02} \cdot y_3, \end{aligned}$$

where byte multiplications are performed in $\mathrm{GF}(2^8)$ defined by the primitive polynomial $z^8 + z^4 + z^3 + z + 1$.

2.3.4 The I Function

The I function accepts a 128-bit word $x = x_0 \| x_1 \| x_2 \| x_3$ as input which consists of the concatenation of four 32-bit words x_i. The I function comprises the application of a byte permutation, followed by an MDS layer and then the same byte permutation. The MDS layer consists of a single 4×4 involutive MDS matrix M' which is applied to the input four times in parallel. The output of the I function is a 128-bit output word $y = y_0 \| y_1 \| y_2 \| y_3$. The MDS matrix M' is provided below.

$$M' = \begin{pmatrix} \mathsf{01} & \mathsf{08} & \mathsf{02} & \mathsf{0A} \\ \mathsf{08} & \mathsf{01} & \mathsf{0A} & \mathsf{02} \\ \mathsf{02} & \mathsf{0A} & \mathsf{01} & \mathsf{08} \\ \mathsf{0A} & \mathsf{02} & \mathsf{08} & \mathsf{01} \end{pmatrix}.$$

The application of I to x is denoted by

$$y_0 \| y_1 \| y_2 \| y_3 = I(x_0 \| x_1 \| x_2 \| x_3).$$

Let $x_i = x_{i,0} \| x_{i,1} \| x_{i,2} \| x_{i,3}$ and $y_i = y_{i,0} \| y_{i,1} \| y_{i,2} \| y_{i,3}$ be further broken into sequences of four 8-bit words $x_{i,j}$ and $y_{i,j}$, respectively. The application of the I function can be

written as follows for $j = 0, 1, \ldots, 3$

$$\begin{aligned}
y_{0,j} &= \mathtt{01} \cdot x_{0,j} \oplus \mathtt{08} \cdot x_{1,j} \oplus \mathtt{02} \cdot x_{2,j} \oplus \mathtt{0A} \cdot x_{3,j},\\
y_{1,j} &= \mathtt{08} \cdot x_{0,j} \oplus \mathtt{01} \cdot x_{1,j} \oplus \mathtt{0A} \cdot x_{2,j} \oplus \mathtt{02} \cdot x_{3,j},\\
y_{2,j} &= \mathtt{02} \cdot x_{0,j} \oplus \mathtt{0A} \cdot x_{1,j} \oplus \mathtt{01} \cdot x_{2,j} \oplus \mathtt{08} \cdot x_{3,j},\\
y_{3,j} &= \mathtt{0A} \cdot x_{0,j} \oplus \mathtt{02} \cdot x_{1,j} \oplus \mathtt{08} \cdot x_{2,j} \oplus \mathtt{01} \cdot x_{3,j},
\end{aligned}$$

where byte multiplications are performed in $\mathrm{GF}(2^8)$ defined by the primitive polynomial $z^8 + z^4 + z^3 + z^2 + 1$. As can be carefully observed, each matrix takes one byte from each branch as input, which is clearly depicted in Figure 1(c). The output of I can be written as

$$y_{0,0} \| \ldots \| y_{0,3} \| y_{1,0} \| \ldots \| y_{3,3} = I(x_{0,0} \| \ldots \| x_{0,3} \| x_{1,0} \| \ldots \| x_{3,3}).$$

Listing 1: A high-level description of Durian

```
Durian(state, masterkey) {
  KeySchedule(masterkey, whiteningkey, roundsubkey);

  // XOR with pre-whitening keys
  for (j=0; j<4; j++) state[j] = state[j]^whiteningkey[j];

  for (i=1; i<=r; i++) RoundA(state, roundsubkey, i);

  I(state);

  for (i=r+1; i<=R; i++) RoundB(state, roundsubkey, i);

  // XOR with post-whitening keys
  for (j=0; j<4; j++) state[j] = state[j]^whiteningkey[j+4];
}
```

Listing 2: Durian components

```
RoundA(state, roundsubkey, i) {
  x[0] = F(roundsubkey[2*i-2], state[0])^state[1];
  x[1] = state[2];
  x[2] = F(roundsubkey[2*i-1], state[2])^state[3];
  x[3] = state[0];

  if (i < r) {
    for (j=0; j<4; j++) state[j] = x[j];
  }
  else {
    for (j=0; j<4; j++) state[j] = x[(j+3) % 4];
  }
}

RoundB(state, roundsubkey, i) {
  x[0] = F(roundsubkey[2*i-1], state[2])^state[3];
  x[1] = state[0];
  x[2] = F(roundsubkey[2*i-2], state[0])^state[1];
  x[3] = state[2];

  if (i < R) {
    for (j=0; j<4; j++) state[j] = x[j];
  }
  else {
    for (j=0; j<4; j++) state[j] = x[(j+1) % 4];
  }
}

F(subkey, x) {
```

```
30      x = x^subkey;
31      x = S(x);
32      x = P(x);
33    }
```

3 Processing the Secret Key Block

The lengths of acceptable master key $\breve{k} = \breve{k}_0 \| \breve{k}_1 \| \ldots \| \breve{k}_{(|\breve{k}|/32)-1}$ are 128, 192 or 256 bits for Durian-128, Durian-192 and Durian-256, respectively. The key schedule produces a set of eight 32-bit whitening keys wk_i and $2\breve{R}$ 32-bit round subkeys rk_i. These keys are produced as follows and the algorithm is given in pseudocode form in Listing 3.

1. If the length of the master key is less than 256 bits, then pad the rightmost bit with bit '1', followed by as many bit '0' until the length is 256 bits. If the master key is already 256 bits, then use the master key as it is. Let this 256-bit value be denoted by $mk = mk_0 \| mk_1 \| \ldots \| mk_7$ which consists of a sequence of eight 32-bit words mk_i. As an example, for Durian-128, $mk_i = \breve{k}_i$ for $i = 0, 1, \ldots, 3$, $mk_4 =$ `80000000` and $mk_i =$ `0` for $i = 5, 6, 7$.

2. Initialize two 32-bit round subkey constants rc_0, rc_1 which are given in Table 2. The values are unique for different variants of Durian.

3. Setup an array of 32-bit registers t_i where the master key words are used as the initial values as follows
$$t_{i-8} = mk_i$$
for $i = 0, 1, \ldots, 7$.

4. The registers are then updated using the following relation, where its maximum iteration is related to the number of rounds, i.e. $i = 0, 1, \ldots, 2\breve{R} + 19$:
$$t_i = \begin{cases} S((t_{i-8} \oplus t_{i-6} \oplus t_{i-5} \oplus t_{i-1} \oplus rc_0 \oplus i) \lll 11) & \text{if } i \bmod 4 = 0, \\ (t_{i-8} \oplus t_{i-6} \oplus t_{i-5} \oplus t_{i-1} \oplus rc_1) \lll 7 & \text{otherwise.} \end{cases}$$

5. The round subkeys are derived as follows which starts at t_{12}.

   (a) The first part of the whitening keys are:
   $$wk_i = t_{i+12}$$
   for $i = 0, 1, \ldots, 3$.

   (b) The round subkeys are:
   $$rk_i = t_{i+16}$$
   for $i = 0, 1, \ldots, 2\breve{R} - 1$.

   (c) The second part of the whitening keys are:
   $$wk_{i+4} = t_{i+16+2\breve{R}}$$
   for $i = 0, 1, \ldots, 3$.

   The complete equations for the registers t_i are given in the full version of this article.

Table 2: Round subkey constants

Key length (in bits)	rc_0	rc_1
128	517CC1B7	27220A94
192	2066EFA6	0C4794C2
256	A62066EF	C20C4794

The round subkey constants for Durian-192 and Durian-256 can be derived from Durian-128 as follows:

$$rc_0 = S(\texttt{517CC1B7} \ggg 7), \quad rc_1 = S(\texttt{27220A94} \ggg 11) \quad \text{for Durian} - 192,$$
$$rc_0 = S(\texttt{517CC1B7} \ggg 15), \quad rc_1 = S(\texttt{27220A94} \ggg 19) \quad \text{for Durian} - 256.$$

Listing 3: The key scheduling algorithm

```
KeySchedule(masterkey, whiteningkey, roundsubkey) {
  pad(masterkey); // pad the key if necessary

  for (i=0; i<8; i++) t[i-8] = masterkey[i];

  for (i=0; i<2*R+20; i++) {
    if ((i % 4) == 0)
      t[i] = S((t[i-8]^t[i-6]^t[i-5]^t[i-1]^rc0^i) <<< 11);
    else
      t[i] = (t[i-8]^t[i-6]^t[i-5]^t[i-1]^rc1) <<< 7;
  }

  // 1st part of the whitening keys
  for (i=0; i<4; i++) whiteningkey[i] = t[i+12];

  // round subkeys
  for (i=0; i<2*R; i++) roundsubkey[i] = t[i+16];

  // 2nd part of the whitening keys
  for (i=0; i<4; i++) whiteningkey[i+4] = t[i+16+2*R];
}
```

4 Cryptanalysis

This section analyses the block cipher Durian against existing cryptanalytic attacks.

4.1 Preliminaries

Most of the attacks presented in this section is applicable to *round-reduced* versions of Durian. For instance, if an attack is applicable to 7 rounds of Durian-128, then the attack is not applicable to the full 16 rounds of Durian-128. Although the attack is theoretical, it is accepted in the cryptographic literature.

4.2 Differential and Linear Cryptanalysis

Differential cryptanalysis [7, 8] is a chosen plaintext attack that exploits the existence of a linear relationship between the encryption of two plaintexts using the same secret master key. Given a pair of inputs to a block cipher with a certain input difference, the output difference after some number of rounds can be predicted with high probability.

The expected difference at this round is used in a key recovery attack to identify possible subkey bits.

Linear cryptanalysis is a known plaintext attack introduced by Matsui in 1993 [33]. In a basic linear attack, a one-bit linear relationship between selected bits of the input, output and master key of the block cipher is constructed which is valid with high probability. These relations are used in a key recovery attack to identify possible subkey bits.

In order to prove the resistance of Durian against differential and linear cryptanalysis, we adopted the number of active S-boxes approach. This approach was also used to evaluate the security of the AES against these attacks [21, Section 9]. As mentioned by Bogdanov and Shibutani [14], since differential and linear cryptanalysis are closely related, it is assumed that the resistance of Durian against these attacks using the number of active S-boxes approach is the same.

The method to count the number of active S-boxes used here is based on the propagation of difference weights, as used in Shirai and Shibutani [40]. Recall that the F function used in Durian is the same in all rounds and hence, difference cancellations may occur in the difference propagation. This is explained as follows. Let $x = x_0||x_1||\ldots||x_{m-1}$ compose of the concatenation of m n-bit words. The Hamming weight $h_w(x)$ is defined as

$$h_w(x) = \#\{i \mid 0 \leq i \leq (m-1),\ x_i \neq 0\}.$$

In order words, the Hamming weight of x is the number of non-zero n-bit words in x.

Let $\Delta_{w_{2i-2}}, \Delta_{z_{2i-2}}$ and $\Delta_{w_{2i-1}}, \Delta_{z_{2i-1}}$ denote the pair of input and output differences of the two F functions in round i where $1 \leq i \leq \breve{R}$. The input difference to the F function in the fourth round is cancelled if the input difference to specific F functions in the first and third rounds are the same. Specifically,

$$h_w(\Delta_{w_{2i+4}}) = 0$$

if $h_w(\Delta_{w_{2i+2}}) = h_w(\Delta_{w_{2i-1}})$ and $h_w(\Delta_{w_{2i+1}}) - h_w(\Delta_{z_{2i-1}}) = 0$. This 3-round difference cancellation is depicted in Figure 2. The same difference cancellation also occurs for $\Delta_{w_{2i+3}}$.

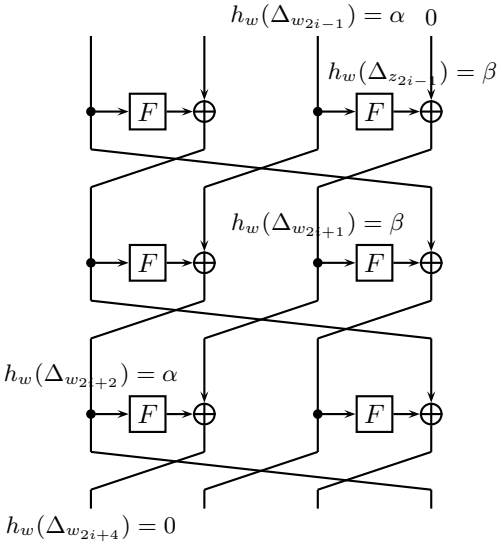


Figure 2: A 3-round difference cancellation

Let $(\Delta_{\breve{x}_{4(i-1)}}, \Delta_{\breve{x}_{4(i-1)+2}})$ and $(\Delta_{\breve{y}_{4(j-1)}}, \Delta_{\breve{y}_{4(j-1)+2}})$ denote the input differences to the F function in rounds $i = 1, 2, \ldots, r$ and $j = r+1, r+2, \ldots, R$, respectively. Furthermore,

$\Delta_{\breve{x}_i} = (\Delta_{\breve{x}_{i,0}}, \Delta_{\breve{x}_{i,1}}, \Delta_{\breve{x}_{i,2}}, \Delta_{\breve{x}_{i,3}})$ and $\Delta_{\breve{y}_i} = (\Delta_{\breve{y}_{i,0}}, \Delta_{\breve{y}_{i,1}}, \Delta_{\breve{y}_{i,2}}, \Delta_{\breve{y}_{i,3}})$ consist of the concatenation of four 8-bit words $\Delta_{\breve{x}_{i,j}}$ and $\Delta_{\breve{y}_{i,j}}$, respectively where $r = R/2$. Therefore the Hamming weights $h_w(\Delta_{\breve{x}_i})$ and $h_w(\Delta_{\breve{y}_i})$ do not exceed 4, i.e. $0 \leq h_w(\Delta_{\breve{x}_i}) \leq 4$ and $0 \leq h_w(\Delta_{\breve{y}_i}) \leq 4$. The minimum number of active S-boxes for an R-round Durian is counted using the following formulae:

$$\min \left\{ \sum_{i=0}^{r-1} \left(h_w(\Delta_{\breve{x}_{4i}}) + h_w(\Delta_{\breve{x}_{4i+2}}) \right) + \sum_{i=r}^{R-1} \left(h_w(\Delta_{\breve{y}_{4i}}) + h_w(\Delta_{\breve{y}_{4i+2}}) \right) \right\}.$$

Table 3 compares the minimum number of differentially active S-boxes between traditional 4-branch generalized Feistel network (GFN)[2] denoted GFN_4, CLEFIA (which employs the Diffusion Switching Mechanism (DSM) technique [39, 41]), $\text{GFN}_4^{\text{SPS}}$ which is a GFN_4 that uses a SPS F function [13, 14], and Durian. It can be seen that, after 7 rounds, the number of differentially active S-boxes of Durian is superior to that of GFN_4. It is also comparable to the DSM technique used in CLEFIA which uses two distinct F-functions in each round. Durian uses only a single F function.

For a block size of 128 bits, the number active S-boxes n_a needed for a cipher to be secure against differential and linear cryptanalysis should satisfy

$$p^{n_a} < 2^{-128}$$

where $p = DP_{\max} = 2^{-5.4}$ for differential cryptanalysis and $p = LP_{\max} = 2^{-6}$ for linear cryptanalysis. Therefore, $n_a \geq 24$ for differential cryptanalysis and $n_a \geq 22$ for linear cryptanalysis. According to Table 3, the minimum number of rounds that is sufficient to protect Durian against both attacks is 12. At 12 rounds, there are a minimum of 30 differentially active S-boxes. Since the smallest variant of this cipher has 16 rounds, we believe that it provides ample protection against these two basic attacks.

In a related work, Knudsen [29] reported a 12-round iterative characteristic on an early version of Durian without the I function. It is formed by iterating a 6-round characteristic with probability p^{12} that has at least 12 active S-boxes. As stated previously, $p = 2^{-5.4}$. The 12-round characteristic thus has a probability of $(2^{-5.4})^{24} \approx 2^{130}$ which prohibits the execution of the attack. Knudsen [29] added that the addition of the I function would bring down this probability even further, though only slightly. This is in line with our previous analysis that sets a minimum of 12 rounds for the cipher to be secure against traditional differential cryptanalysis.

We have also analysed and argued the security of Durian against boomerang [44], amplified boomerang [24], impossible differential [6], integral [28], algebraic [18, 19], related-key [5, 27, 25, 26], biclique [11, 12], slide-based [9, 10] and known-key [30] attacks. Due to length-limitation, these analyses are available at the full version of this article.

5 Design Rationale

This section describes the reasons behind the design choices for the block cipher Durian.

5.1 Block and Key Sizes

The block cipher Durian accepts a 128-bit plaintext block and key lengths of 128, 192 and 256 bits. These parameters are consistent with the Advanced Encryption Standard (AES) requirements [36, 21] and are commonly supported by other ciphers. Extending the key size of Durian to a size larger than 256 bits would require a longer array of registers

[2] A traditional 4-branch GFN is composed of four data lines (as in Durian) and all round functions are identical.

Table 3: Minimum number of differentially active S-boxes in r rounds

r	GFN_4	CLEFIA [43]	$\text{GFN}_4^{\text{SPS}}$ [13, 14]	Durian
1	0	0	0	-
2	1	1	5	1
3	2	2	10	-
4	6	6	15	4
5	8	8	20	-
6	12	12	30	8
7	12	14	30	-
8	13	18	35	15
9	14	20	40	-
10	18	22	45	20
11	20	24	50	-
12	24	28	60	30
13	24	30	60	-
14	25	34	65	34
15	26	36	70	-
16	30	38	75	40
17	32	40	80	-
18	36	44	90	40
19	36	46	90	-
20	37	50	95	45
21	38	52	100	-
22	42	55	105	50
23	44	56	110	-
24	48	59	120	60
25	48	62	120	-
26	49	65	125	65

and consequently a different underlying primitive polynomial to update the value of the registers. It is possible to expand the 128-bit block size in steps of 64 bits, which would require a different byte permutation in the I function in the middle of the cipher.

5.2 Number of Rounds

The number of rounds (16, 20 and 24) for each variant of the cipher is determined based on the security analysis provided in Section 4. In particular, it was shown in Section 4.2 that constructing a differential characteristic that covers 12 or more rounds is not useful as the probability of such differentials is significantly low and thus, not useful in an attack. The increase of the number of rounds in steps of 4 can be regarded as imitating the AES. In AES, the increase is in steps of 2, which can arguably be said to equal 4-round Durian since 1 round of AES updates the whole state while Durian requires 2 rounds to accomplish the same.

5.3 The Structure

The structure of Durian is based on an extended variant of the Feistel construction [22], widely used and analyzed over the past 40 years, with a comprehensive survey provided by Nachef, Patarin, and Volte [34]. Specifically, Durian employs a modified 4-branch Type-2 generalized Feistel network (GFN) [45], allowing identical encryption and decryption circuits, differing only in the order of round subkeys, thus termed *involutional*. In contrast,

the block cipher CLEFIA [42], which uses the standard 4-branch Type-2 GFN, requires different fixed rotation offsets between rounds for encryption and decryption[3]. While the rotations may have minimal hardware cost, they necessitate different encryption and decryption circuits at the algorithm level in CLEFIA.

Several ciphers achieve the involutive property using different methods. Noekeon [20], Anubis [2], and Khazad [3] employ involutive components like S-boxes and diffusion layers. The Data Encryption Standard (DES) is also involutive due to its use of the Feistel network. The lightweight block ciphers PRINCE [15] and PRINCE v2 [17] achieve this property by incorporating an involutive component in the middle and making the second half the inverse of the first, classified as reflection ciphers [16]. Beyne and Chen have further validated the security of these designs [4]. The structure of DURIAN is inspired by this approach.

There are some notable differences between Durian and PRINCE. One obvious difference is the use of the SPN structure in PRINCE. Apart from this, PRINCE uses non-involutive components in the first half of the cipher and thus, requires the inverse of the components in the second half of the cipher. This is to preserve the involutive structure of the cipher. In contrast, the only difference between the first and second half of Durian is the rotation between the rounds. The nonlinear and linear diffusion transformations in the F function need not be involutive due to the inherent property of the GFN.

The modified GFN structure also has other advantages in terms of security. In particular, the resistance of the cipher against differential and linear cryptanalysis is improved compared to the traditional Type-2 GFN. Furthermore, the resistance is comparable to the diffusion switching mechanism technique employed in CLEFIA [41]. This would be further explained in Section 4.2.

5.4 The Linear Diffusion Transformation

The linear diffusion transformations in the F and I functions of Durian use the maximal distance separable (MDS) matrix. The main advantage of using an MDS matrix is that the corresponding transformation has maximal *branch number* [21]. The branch number denotes the minimum number of nonzero input and output words of the function represented by the matrix. This number is used to estimate the strength of a block cipher against differential and linear cryptanalysis.

Definition 1. Let $x = (x_0, x_1, \ldots, x_{m-1})$ compose of the concatenation of m n-bit words. The Hamming weight $h_w(x)$ is defined as

$$h_w(x) = \#\{i \mid 0 \leq i < m,\ x_i \neq 0\}.$$

In order words, the Hamming weight of x is the number of non-zero n-bit words in x.

Definition 2. Let L denote a linear diffusion transformation that accepts and outputs m n-bit words. Its branch number, denoted $\mathcal{B}(L)$, is defined as

$$\mathcal{B}(L) = \min\{h_w(x) + h_w(L(x))\}$$

where $x \neq 0$ and $\mathcal{B}(L) \leq m + 1$.

In essence, the branch number denotes the minimum number of non-zero input and output words of the linear transformation L, given a non-zero input.

The linear diffusion transformations used in the F and I functions (see Sections 2.3.3 and 2.3.4) are taken from the AES and CLEFIA block ciphers, respectively. Both possess the maximal branch number of $\mathcal{B}(L) = 5$ for $m = 4$ and $n = 8$.

[3]Encryption requires 32-bit left rotation, while decryption needs 32-bit right rotation.

5.5 The Nonlinear Transformation

In Durian, the 8×8 bijective S-box s in S is the only component that provides nonlinearity in the cipher. The S-box is constructed from a non-permutation power function that provides high nonlinearity and low differential uniformity. Since the outputs of this function is a non-permutation, additional operations are required to be made to the outputs in order to convert the non-permutation outputs into a permutation.

The non-permutation power function used to construct the S-box of Durian is $65z^3 + 64z^4 + 17$ where $z \in \{0,1\}^8$. Using a specialized technique, the outputs of this function are modified to produce a bijective S-box.

The S-box of Durian has the following properties:

- Nonlinearity of 112.
- Algebraic degree of 7.
- Maximum differential probability of $2^{-5.4}$ (i.e. differential uniformity of 6).
- Maximum linear probability of 2^{-6} (i.e. linear approximation of 16).
- No fixed points and opposite fixed points.

The Langrange interpolation and the algebraic normal forms of the S-box are given in the full version of this article.

The maximum differential probability has to be low in order to thwart differential cryptanalysis. This can be derived from the differential probability of the S-box.

Definition 3. Given input and output differences $\Delta_a, \Delta_b \in \{0,1\}^n$, the differential probability of the S-box S for these differences is defined as

$$DP(\Delta_a, \Delta_b) = \frac{\#\{x \in \{0,1\}^n \mid S(x) \oplus S(x \oplus \Delta_a) = \Delta_b\}}{2^n}.$$

Definition 4. The maximum differential probability $DP_{\max}$ for an S-box is given by

$$DP_{\max} = \max_{\Delta_a, \Delta_b \neq 0} DP(\Delta_a, \Delta_b).$$

For Durian,

$$DP_{\max} = \frac{6}{256} = 2^{-5.4}.$$

The above maximum differential probability is used to argue the security of Durian against differential cryptanalysis, which is described in Section 4.2.

As in the differential probability, the maximum linear probability also has to be low in order to thwart linear cryptanalysis. This can be derived from the linear probability of the S-box

Definition 5. Given input and output masks $\lambda_a, \lambda_b \in \{0,1\}^n$, the linear probability of the S-box S for these masks is defined as

$$LP(\lambda_a, \lambda_b) = \left(\frac{\#\{x \in \{0,1\}^n \mid \lambda_a \cdot x = \lambda_b \cdot S(x)\}}{2^{n-1}} - 1 \right)^2.$$

The above can also be defined as

$$LP(\lambda_a, \lambda_b) = (2 \cdot \mathrm{Prob}_x\{\lambda_a \cdot x = \lambda_b \cdot S(x)\} - 1)^2.$$

Definition 6. The maximum linear probability $LP_{\max}$ for an S-box is given by

$$LP_{\max} = \max_{\lambda_a, \lambda_b \neq 0} LP(\lambda_a, \lambda_b).$$

For Durian,

$$LP_{\max} = \left(\frac{16+128}{128} - 1\right)^2 = 2^{-6}$$

The above maximum linear probability is used to argue the security of Durian against linear cryptanalysis, which is described in Section 4.2.

5.6 Key Schedule

The criteria for designing the key schedule is as follows.

1. Supports the same key lengths as the AES, which are 128, 192 and 256 bits. This makes the cipher compatible with existing software and hardware infrastructure.

2. Generating subkeys for 128, 192 and 256 -bit keys should be performed using the same algorithm.

3. The number of different constants to be stored should be kept at a minimum. For all instances of the cipher, for a compact implementation, only 64 bits of constant value are required to be stored. This is because the constants for Durian-192 and Durian-256 variants can be derived from a single fixed 64-bit value used in Durian-128.

4. Knowledge of a pair of 32-bit subkeys in one round does not immediately give the value of the subkey of either the previous or subsequent round. This is to make key recovery attacks much harder. Following Henricksen's classification, Durian's key schedule is of Type 2C in which all master key bits are used to derive each round subkey [23, Section 2.1]. Type 2C is considered to be one of the strongest in this classification.

The key schedule is inspired by the key schedule of the block cipher Serpent [1]. In Serpent, the 256-bit master key is written as eight 32-bit values $w_{i-8}, w_{i-7}, \ldots, w_{i-1}$. Then, the following affine recurrence is run to produce a set of 132 32-bit intermediate key values $w_0, w_1, \ldots, w_{131}$:

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11$$

where $\phi =$ `9E3779B9`. The intermediate values are further subjected to the eight 4×4 S-boxes of Serpent before being used as the 33 128-bit round subkeys.

For ease of comparison, the key schedule of Durian is again given here. It is based on the following relation

$$t_i = \begin{cases} S((t_{i-8} \oplus t_{i-6} \oplus t_{i-5} \oplus t_{i-1} \oplus rc_0 \oplus i) \lll 11) & \text{if } i \bmod 4 = 0, \\ (t_{i-8} \oplus t_{i-6} \oplus t_{i-5} \oplus t_{i-1} \oplus rc_1) \lll 7 & \text{otherwise.} \end{cases}$$

The relation is based on a linear feedback shift register (LFSR) and the underlying polynomial i.e. $z^8 + z^7 + z^3 + z^2 + 1$ is primitive. The different rotation offsets and constants, together with the nonlinear S function, prevents the construction of weak keys and related keys. The key material is derived only after the iteration is performed 12 times and after the nonlinear transformation has been applied four times. This is to reduce the occurrence of trivial difference trails in the round subkeys. The use of the S-box only in the fourth iteration of the above relation is for both nonlinearity and efficiency purposes.

The constants used in the key schedule algorithm are derived from the first 64-bit hexadecimal representation of π^{-1} where π is the mathematical constant that defines the ratio of a circle's circumference to its diameter. The value of the constant is $rc_0 =$ `517CC1B7`

and $rc_1 = \texttt{27220A94}$ for Durian-128. The round subkey constants for Durian-192 and Durian-256 can be derived from Durian-128 as follows:

$$\begin{aligned} rc_0 &= S(\texttt{517CC1B7} \ggg 7), & rc_1 &= S(\texttt{27220A94} \ggg 11) & \text{for Durian} - 192, \\ rc_0 &= S(\texttt{517CC1B7} \ggg 15), & rc_1 &= S(\texttt{27220A94} \ggg 19) & \text{for Durian} - 256. \end{aligned}$$

This is an example of a *nothing up my sleeve* number. The derivation of constants from such numbers is common in cryptography, and is used, for instance, in the block ciphers Blowfish [38], RC5 [37] and ARIA [32].

6 Implementation Aspects

This section describes an optimization technique that can be used to implement Durian. The technique combines the application of the S-boxes in the S function and the MDS matrix in the P function. This technique has been similarly applied to the Sony block cipher CLEFIA [43, Chap. 4] and the AES [21, Chap. 4]. We repeat some of the materials here.

6.1 The F Function

Let (x_0, x_1, x_2, x_3) denote the input of the F function after the key addition and let (y_0, y_1, y_2, y_3) denote the output of the F function. The input and output is related as:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix} \begin{pmatrix} s(x_0) \\ s(x_1) \\ s(x_2) \\ s(x_3) \end{pmatrix}.$$

The above can be viewed as a linear combination of four column vectors:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} \texttt{02} \\ \texttt{01} \\ \texttt{01} \\ \texttt{03} \end{pmatrix} s(x_0) \oplus \begin{pmatrix} \texttt{03} \\ \texttt{02} \\ \texttt{01} \\ \texttt{01} \end{pmatrix} s(x_1) \oplus \begin{pmatrix} \texttt{01} \\ \texttt{03} \\ \texttt{02} \\ \texttt{01} \end{pmatrix} s(x_2) \oplus \begin{pmatrix} \texttt{01} \\ \texttt{01} \\ \texttt{03} \\ \texttt{02} \end{pmatrix} s(x_3). \quad (1)$$

Then, the following tables T_i can be defined where the input size is 8 bits and the output size is 32 bits

$$T_0(x) = \begin{pmatrix} \texttt{02} \cdot s(x) \\ \texttt{01} \cdot s(x) \\ \texttt{01} \cdot s(x) \\ \texttt{03} \cdot s(x) \end{pmatrix}, \qquad T_1(x) = \begin{pmatrix} \texttt{03} \cdot s(x) \\ \texttt{02} \cdot s(x) \\ \texttt{01} \cdot s(x) \\ \texttt{01} \cdot s(x) \end{pmatrix},$$

$$T_2(x) = \begin{pmatrix} \texttt{01} \cdot s(x) \\ \texttt{03} \cdot s(x) \\ \texttt{02} \cdot s(x) \\ \texttt{01} \cdot s(x) \end{pmatrix}, \qquad T_3(x) = \begin{pmatrix} \texttt{01} \cdot s(x) \\ \texttt{01} \cdot s(x) \\ \texttt{03} \cdot s(x) \\ \texttt{02} \cdot s(x) \end{pmatrix}.$$

Therefore, Equation 1 can be rewritten as

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = T_0(x_0) \oplus T_1(x_1) \oplus T_2(x_2) \oplus T_3(x_3).$$

Each table T_i has 256 4-byte entries and thus, one table requires $256 \times 4 = 2^8 \times 2^2 = 2^{10}$ bytes, or 1KB. There are four tables and hence, 4KB of storage is required to implement this technique. Note that the tables T_1, T_2 and T_3 are rotated versions of T_0. A further reduction of storage is thus possible by only implementing T_0. Using this technique, the storage requirement is just 1KB. In matrix form, we have

$$T_1(x) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot T_0(x), \qquad T_2(x) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cdot T_0(x),$$

$$T_3(x) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot T_0(x).$$

Note however, in exchange for saving the cost of storage, the implementation requires additional rotation operations. The values for the tables T_0, T_1, T_2 and T_3 are given in the full version of this article.

6.2 The I Function

The I function contains an involutive MDS matrix and thus, the same argument described in the previous section can be used here. The exception is that there is no application of the S-box. Let $(x_{0,0}, x_{0,1}, \ldots, x_{0,3}, x_{1,0}, \ldots, x_{1,3}, x_{2,0}, \ldots x_{3,3})$ denote the 16-byte input of the function θ and $(y_{0,0}, y_{0,1}, \ldots, y_{0,3}, y_{1,0}, \ldots, y_{1,3}, y_{2,0}, \ldots, y_{3,3})$ denote its output. The output of θ is related to its input by

$$\begin{pmatrix} y_{0,i} \\ y_{1,i} \\ y_{2,i} \\ y_{3,i} \end{pmatrix} = \begin{pmatrix} \texttt{01} & \texttt{08} & \texttt{02} & \texttt{0A} \\ \texttt{08} & \texttt{01} & \texttt{0A} & \texttt{02} \\ \texttt{02} & \texttt{0A} & \texttt{01} & \texttt{08} \\ \texttt{0A} & \texttt{02} & \texttt{08} & \texttt{01} \end{pmatrix} \begin{pmatrix} x_{0,i} \\ x_{1,i} \\ x_{2,i} \\ x_{3,i} \end{pmatrix}$$

where $i = 0, 1, 2, 3$.

A similar equation to that of Equation 1 can be built and thus the following tables T_4, T_5, T_6, T_7 can be defined to implement the function θ.

$$T_4(x) = \begin{pmatrix} \texttt{01} \cdot x \\ \texttt{08} \cdot x \\ \texttt{02} \cdot x \\ \texttt{0A} \cdot x \end{pmatrix}, \qquad T_5(x) = \begin{pmatrix} \texttt{08} \cdot x \\ \texttt{01} \cdot x \\ \texttt{0A} \cdot x \\ \texttt{02} \cdot x \end{pmatrix},$$

$$T_6(x) = \begin{pmatrix} \texttt{02} \cdot x \\ \texttt{0A} \cdot x \\ \texttt{01} \cdot x \\ \texttt{08} \cdot x \end{pmatrix}, \qquad T_7(x) = \begin{pmatrix} \texttt{0A} \cdot x \\ \texttt{02} \cdot x \\ \texttt{08} \cdot x \\ \texttt{01} \cdot x \end{pmatrix}.$$

The amount of storage required is 4KB. However, similar as before, the tables T_5, T_6 and T_7 can be derived from T_4. So, if this fact is used, only 1KB is required but additional rotation operations need to be performed. The values for the tables T_4, T_5, T_6 and T_7 are given in the full version of this article.

7 Performance Analysis

This section presents a preliminary performance analysis of Durian on both software and hardware. For both platforms, the results were benchmarked with the AES.

7.1 Software

The block ciphers are implemented in the C language and were benchmarked on a 64-bit Intel Xeon 2.93 GHz PC with 48 GB of RAM using Windows 7 Professional. A comparison is performed with an implementation of the AES block cipher in the LibTomCrypt version 1.17 library. All codes are compiled with the optimization flag set to '`-O1`'.

The performance tests were performed to assess the speed of Durian and AES for encryption and the key scheduling algorithm (which produces the subkeys for both encryption and decryption). For encryption, the results were obtained by encrypting 10 MB, 100 MB, 500 MB and 1 GB of data in the electronic codebook (ECB) mode of operation (i.e. the encryption is performed block by block). The performance of the key scheduling algorithm was assessed based on the number keys processed. The results for each of these tests are compiled in Tables 4 and 5, respectively. Note that the figures obtained may not represent the best speed obtained for both Durian and AES. This is to state that there may exist better implementation of these ciphers compared to our implementation.

Table 4: Encryption speed (in seconds) of Durian and AES

Cipher	Data Size				Avg. Throughput
	10 MB	100 MB	500 MB	1 GB	(MBps)
Durian-128	0.075	0.742	3.710	7.406	134.475
AES-128	0.075	0.740	3.701	7.399	134.680
Durian-192	0.090	0.892	4.436	8.880	112.136
AES-192	0.087	0.864	4.300	8.597	115.820
Durian-256	0.104	1.036	5.176	10.371	96.425
AES-256	0.100	0.992	4.942	9.875	100.811

Based on our implementation for encryption, Durian-128 is able to perform encryption at the rate of 134.475 Megabytes per second (MBps). This is slightly slower (about 0.2% slower) than AES-128 which is able to perform encryption at the rate of 134.680 MBps. Durian-192 and Durian-256 are able to encrypt at the rate of 112.136 and 96.425 MBps, respectively. These figures are 3.2% slower than AES-192 which is able to encrypt at 115.820 MBps and 4.4% slower than AES-256 which is able to encrypt at 100.811 MBps. Although the results show that the performance of Durian is slightly slower than the AES, it is however still competitive and comparable to the AES.

Table 5: Key scheduling speed (in seconds) of Durian and AES

Cipher	Number of keys (in millions)				Avg. Throughput
	0.1 M	1 M	10 M	100 M	(M keys per second)
Durian-128	0.016	0.157	1.573	15.736	6.333
AES-128	0.019	0.186	1.842	18.435	5.373
Durian-192	0.017	0.182	1.801	18.048	5.618
AES-192	0.020	0.199	1.961	19.647	5.053
Durian-256	0.021	0.209	2.037	20.375	4.841
AES-256	0.025	0.243	2.411	24.145	4.101

In the key scheduling algorithm, Durian-128 is able to process 6.333 million keys per second. This is considerably faster (about 17.9% faster) than AES-128 which is able to process only 5.373 million keys per second. Durian-192 and Durian-256 are able to process 5.618 and 4.841 million keys per second, respectively. These figures are 11.2% and 18% faster than AES-192 and AES-256, respectively. These results put Durian in a superior

position to that of the AES in processing the secret master key. It is therefore advantageous to implement Durian in systems where the secret key is frequently changed. We would like to stress again that the results obtained are based on our own implementation and may not represent the best speed for Durian. However, the results are sufficient to show the average performance of Durian as compared to the AES.

7.2 Hardware

In this section, we demonstrate the performance of Durian and AES on four FPGA devices from the Xilinx family: Virtex-7, Artix-7, Kintex-7 and Spartan-6. A round-based architecture of the ciphers was utilized. The S-box was implemented as a lookup table and the subkeys were pre-computed. The implementation also made use of 128-bit data path. The algorithms are coded using Verilog and Mentor Graphics ModelSim and Xilinx ISE were used for simulation and design synthesis, respectively.

The results are given in Table 6. It is interesting to note that the results for both Virtex-7 and Kintex-7 are identical. In terms of area, each of the Durian variants occupies significantly fewer slices (between 40%–45%) against its AES counterpart on all platforms. Its throughput, however, cannot seem to match the AES. If we consider the efficiency metric, the Durian variants are slightly inefficient, between 10%–28%, against the AES. Based on this, it can be stated that Durian is still comparable to the AES.

Table 6: Performance of Durian and AES on FPGA

Cipher	Device	Slices	Freq. (MHz)	Throughput (Gbps)	Efficiency (Mbps/slice)
Durian-128	Virtex-7	1657	224.02	1.37	0.82
	Artix-7	1660	170.30	1.04	0.63
	Kintex-7	1657	224.02	1.37	0.82
	Spartan-6	1649	89.12	0.54	0.33
Durian-192	Virtex-7	1658	225.38	1.15	0.70
	Artix-7	1668	171.71	0.88	0.53
	Kintex-7	1658	225.38	1.15	0.70
	Spartan-6	1683	89.66	0.46	0.27
Durian-256	Virtex-7	1725	224.42	0.99	0.57
	Artix-7	1732	171.71	0.76	0.44
	Kintex-7	1725	224.42	0.99	0.57
	Spartan-6	1721	89.66	0.40	0.23
AES-128	Virtex-7	2884	332.82	3.28	1.14
	Artix-7	2731	231.59	2.28	0.83
	Kintex-7	2884	332.82	3.28	1.14
	Spartan-6	2994	126.25	1.24	0.42
AES-192	Virtex-7	2899	292.79	2.50	0.86
	Artix-7	2906	202.62	1.73	0.59
	Kintex-7	2899	292.79	2.50	0.86
	Spartan-6	3082	107.39	0.92	0.30
AES-256	Virtex-7	2997	296.70	2.23	0.75
	Artix-7	2895	202.62	1.53	0.53
	Kintex-7	2997	296.70	2.23	0.75
	Spartan-6	3138	106.70	0.80	0.26

8 Conclusion

Durian is a general-purpose block cipher designed to be comparable to the industry standard AES. It supports a 128-bit message block and offers key options of 128, 192 and 256 bits. Durian achieves the reflective property by combining two Type-2 generalised Feistel networks with an involution function. The primary difference between the first and second halves of the cipher is the direction of rotation, which incurs no additional hardware cost. Given that Durian is a relatively new block cipher, we encourage third-party analysis to thoroughly evaluate and ensure its security.

References

[1] Ross Anderson, Eli Biham, and Lars Knudsen. Serpent: A Proposal for the Advanced Encryption Standard. NIST AES Proposal, August 1998. Available at http://www.cl.cam.ac.uk/~rja14/serpent.html.

[2] Paulo S.L.M. Barreto and Vincent Rijmen. The Anubis Block Cipher. First Open NESSIE Workshop, November 2000. Available at https://www.cosic.esat.kuleuven.be/nessie/workshop/.

[3] Paulo S.L.M. Barreto and Vincent Rijmen. The Khazad Legacy-Level Block Cipher. First Open NESSIE Workshop, November 2000. Available at https://www.cosic.esat.kuleuven.be/nessie/workshop/.

[4] Tim Beyne and Yu Long Chen. Provably Secure Reflection Ciphers. In Yevgeniv Dodis and Thomas Shrimpton, editors, *Advances in Cryptology — CRYPTO 2020, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 234–263. Springer-Verlag, 2020.

[5] Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer-Verlag, 1994.

[6] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer-Verlag, 1999.

[7] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-Like Cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[8] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.

[9] Alex Biryukov. Analysis of Involutional Ciphers: Khazad and Anubis. In Thomas Johansson, editor, *Fast Software Encryption: 10th International Workshop, FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 45–53. Springer-Verlag, 2003.

[10] Alex Biryukov and David Wagner. Advanced Slide Attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer-Verlag, 2000.

[11] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. Cryptology ePrint Archive, Report 2011/449, August 2011. Available at http://eprint.iacr.org/2011/449/.

[12] Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique Cryptanalysis of the Full AES. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer-Verlag, 2011.

[13] Andrey Bogdanov and Kyoji Shibutani. Double SP-Functions: Enhanced Generalized Feistel Networks. In Udaya Parampalli and Philip Hawkes, editors, *Information Security and Privacy, 16th Australasian Conference, ACISP 2011*, volume 6812 of *Lecture Notes in Computer Science*, pages 106–119. Springer-Verlag, 2011.

[14] Andrey Bogdanov and Kyoji Shibutani. Generalized Feistel Networks Revisited. *Designs, Codes and Cryptography*, 66(1–3):75–97, 2013.

[15] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer-Verlag, 2012.

[16] Christina Boura, Anne Canteaut, Lars R. Knudsen, and Gregor Leander. Reflection Ciphers. *Designs, Codes and Cryptography*, 82:3–25, 2017.

[17] Dušan Božilov, Maria Eichlseder, Miroslav Knežević, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. PRINCEv2: More Security for (Almost) No Overhead. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography – SAC 2020*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer-Verlag, 2021.

[18] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.

[19] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, November 2002. Available at http://eprint.iacr.org/2002/044/.

[20] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie Proposal: NOEKEON. First Open NESSIE Workshop, November 2000. Available at http://gro.noekeon.org/.

[21] Joan Daemen and Vincent Rijmen. *The Design of Rijndael, AES – The Advanced Encryption Standard*. Springer-Verlag, 2002.

[22] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, May 1973.

[23] Matt Henricksen. *Design, Implementation and Cryptanalysis of Modern Symmetric Ciphers*. PhD thesis, Queensland University of Technology, June 2005.

[24] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *Fast Software Encryption – FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer-Verlag, 2001.

[25] John Kelsey, Bruce Schneier, and David Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 1996.

[26] John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communication Security, ICICS'97*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer-Verlag, 1997.

[27] Lars Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology – ASIACRYPT '92,*, volume 718 of *Lecture Notes in Computer Science*, pages 22–35. Springer-Verlag, 1993.

[28] Lars Knudsen and David Wagner. Integral Cryptanalysis. In Joan Daeman and Vincent Rijmen, editors, *Fast Software Encryption: 9th International Workshop, FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer-Verlag, 2002.

[29] Lars R. Knudsen. Evaluation of Security Level of Mi-Crypt. Unpublished Report, 2014.

[30] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324. Springer-Verlag, 2007.

[31] Ted Krovetz and Phillip Rogaway. OCB (v1.1). CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2016. https://competitions.cr.yp.to/round3/ocbv11.pdf.

[32] Daesung Kwon, Jaesung Kim, Sangwoo Park, Soo Hak Sung, Yaekwon Sohn, Jung Hwan Song, Yongjin Yeom, E-Joong Yoon, Sangjin Lee, Jaewon Lee, Seongtaek Chee, Daewan Han, and Jin Hong. New Block Cipher: ARIA. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology – ICISC 2003: 6th International Conference*, volume 2971 of *Lecture Notes in Computer Science*, pages 432–445. Springer-Verlag, 2004.

[33] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

[34] Valerie Nachef, Jacques Patarin, and Emmanuel Volte. *Feistel Ciphers*. Springer, 2017.

[35] National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standards (FIPS) 46, 1977.

[36] National Institute of Standards and Technology. Advanced Encryption Standard. Federal Information Processing Standard (FIPS) 197, November 2001.

[37] Ronald L. Rivest. The RC5 Encryption Algorithm. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer-Verlag, 1995.

[38] Bruce Schneier. Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish). In Ross J. Anderson, editor, *Fast Software Encryption, Cambridge Security Workshop*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer-Verlag, 1994.

[39] Taizo Shirai and Bart Preneel. On Feistel Ciphers Using Optimal Diffusion Mappings Across Multiple Rounds. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security*, volume 3329 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2004.

[40] Taizo Shirai and Kyoji Shibutani. Improving Immunity of Feistel Ciphers against Differential Cryptanalysis by Using Multiple MDS Matrices. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption, FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 260–278. Springer-Verlag, 2004.

[41] Taizo Shirai and Kyoji Shibutani. On Feistel Structures Using a Diffusion Switching Mechanism. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 41–56. Springer-Verlag, 2006.

[42] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit Blockcipher CLEFIA. In Alex Biryukov, editor, *Fast Software Encryption: 14th International Workshop, FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer-Verlag, 2007.

[43] Sony Corporation. The 128-bit Blockcipher CLEFIA Security and Performance Evaluations, June 2007. http://bit.ly/2fSx0sM+.

[44] David Wagner. The Boomerang Attack. In Lars Knudsen, editor, *Fast Software Encryption: 6th International Workshop, FSE'99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer-Verlag, 1999.

[45] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 461–480. Springer-Verlag, 1990.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 140–148.

An Attack on The Diophantine Equation of The RSA Variant

Nurul Nur Hanisah Adenan[1], Muhammad Rezal Kamel Ariffin[1,2] and Zahari Mahad[1]

[1] Institute for Mathematical Research, Universiti Putra Malaysia, Selangor, Malaysia
hanisahadenan@gmail.com,zaharimahad@upm.edu.my
[2] Department of Mathematics and Statistics, Universiti Putra Malaysia, Selangor, Malaysia
rezal@upm.edu.my

Abstract. This paper presents an attack on the standard modulus of RSA $N = pq$ where p, q are two large and balanced primes. Considering the modified Euler Totient function $\psi(N) = (p^2 - 1)(q^2 - 1)$, we showed that the modulus can be factored if there exists the Diophantine equation such that $e_i x - \psi(N_i) y_i = 1$ where e_i, N_i are the multiple public keys, $\psi(N_i), y_i$ are the numerous private keys while x remains the same. Exploiting the information of public keys N_i, e_i and the Diophantine equation, we managed to retrieve p, q via lattice basis reduction method.

Keywords: RSA cryptosystem · Integer factorization problem · Lattice reduction

1 Introduction

The security of digital data sent through unsafe channels relies on the ability of any mechanisms to keep the data unreadable to unauthorized parties. One of the well-known mechanisms that have been used for over 40 years since its invention namely RSA [9] provides security and authenticity of digital data. Furthermore, the hardness of factoring the modulus $N = pq$ such that p, q are large primes, solving key equation $ed - k\phi(N) = 1$ where $d, k, \phi(N)$ are kept secret, and solving $m^e \equiv C \pmod N$ given the known values of e, C, N ensure the security of the RSA against adversaries. Regardless, consistent analyses of finding its weaknesses have been made on this cryptosystem to ensure the RSA remains practicable and impregnable. Thus, it will help cryptographic practitioners avoid using these weaknesses in their algorithms.

To this day, the construction of the RSA algorithms is indeed impenetrable via direct intrusion but not through algebraic analysis. In 1990, Wiener [11] managed to factor the modulus $N = pq$ by utilizing the continued fraction expansion method provided the decryption key d must be $d < \frac{1}{3}N^{\frac{1}{4}}$. Later in 1999, Boneh and Durfee [3] managed to increase the bound up to $d < N^{0.292}$ using the lattice reduction method. Since then, active research has been done either to increase the efficiency or to improve the bound of the private key d that is insecure from attacks.

Numerous modifications have been made to the elements involved in the RSA algorithms that allow for faster key generation and decryption. Multi-power RSA and multi-prime RSA are such instances of RSA variants that alter the modulus $N = pq$ into $N = p^r q^s$ and $N = pqrs$ respectively. Nonetheless, these patterns are also vulnerable if they fall into certain conditions. Apart from that, researchers are also interested in studying and modifying the original Euler Totient Function. Suppose the original function used in the

RSA was $(p-1)(q-1)$. It is proven that this function can be exploited in factoring the modulus N provided some conditions are satisfied. Therefore, the designers focused on modifying the equations in their new cryptosystem to strengthen the security. For instance, in 1995, [4] proposed a modified RSA scheme that was based on singular elliptic curve $y^2 \equiv x^3 + bx^2 \mod N$ where $N = pq$ is the RSA modulus. They used the new equation $\psi(N) = (p^2-1)(q^2-1)$ in their key generation whilst preserving the other original keys. However, the works in [1, 2, 12] showed that this type of RSA variant is vulnerable to certain circumstances.

Another instance of attack upon the RSA is when multiple public keys are utilized without changing the private key. Since the communication occurs between various parties, thus one may attempt to regenerate different pair of public keys but the private key remains the same. In 2014, Nitaj et al. [7] showed that it is vulnerable for one to use either $e_i d - k_i\phi(N_i) = z_i$ or $e_i d_i - k\phi(N_i) = z_i$ provided some conditions are fulfilled. In their work, they focused on the standard modulus $N + pq$. Few years after, in 2018, Rahman et al. [8] proposed an attack on the modulus $N = p^2q$ using the same approach as in [7]. Their attack worked when the primes p_i and q_i shared the most significant bits. In 2024, Ruzai et al. [10] presented an attack on the standard modulus $N = pq$. Focusing on the public key pairs (N_i, e_I) and a fixed value y, they worked on the equation $e_i x_i^2 - y^2\phi(N_i) = z_i$ and showed that the modulus N_i can be factored via lattice reduction technique.

In this paper, we present an attack on the system of equations utilizing modified RSA key equation $\psi(N) = (p^2-1)(q^2-1)$. We showed that using the distinct public keys e, N and private key y is insecure while maintaining the same private exponent x such that $e_i x - \psi(N_i)y_i = 1$. Utilizing lattice reduction method [5], we managed to retrieve p_i and q_i of $N_i = p_i q_i$ for $1 < i < k; k \geq 2$ in polynomial time. This paper is organized as follows. Section 2 describes briefly the mathematical tools that have been used throughout this work. Meanwhile, we present the previous attacks and our new attack in Section 3 respectively. Lastly, we conclude our work in Section 4.

2 Preliminaries

This section presents essential mathematical tools needed throughout our works.

2.1 Lattice and diophantine equations

Let $v_1, v_2, \ldots, v_k \in \mathbb{R}^n$ be k linearly independent vectors with $k \leq n$. Thus, the form of $\mathcal{L} = \left\{\sum_{i=1}^{k} u_i v_i | u_i \in \mathbb{Z}\right\}$ is called a lattice spanned by the basis containing the set of all integer linear combinations of the vectors $v_i, v_2, \ldots, v_k$, and k is its dimension. Meanwhile, the determinant of $\mathcal{L}$ is defined as $\det \mathcal{L} = \sqrt{\det(V^T V)}$ where V is the matrix of the basis vector $v_1, \ldots, v_k$ in $\mathbb{R}^n$. Later on, Lenstra, Lenstra, and Lovász [5] designed an efficient algorithm objectively to find the reduced basis vectors. Their work is presented in the following theorem.

Theorem 1. *Let $\mathcal{L}$ be a lattice spanned by a basis $\{v_1, \ldots, v_k\}$ with k dimension. A reduced basis $\{b_1, \ldots, b_k\}$ is produced via the LLL algorithm satisfying*

$$||b_1|| \leq \cdots ||b_i|| \leq 2^{\frac{k(k-1)}{4(k+1-i)}} \det(\mathcal{L})^{\frac{1}{k+1-i}}.$$

for all $1 \leq i \leq k$.

The LLL algorithm has been implemented abundantly in various applications. For instance, it solved the problem of simultaneous diophantine approximation. The following theorem presented the method described by Lenstra, Lenstra, and Lovász [5] to solve this problem considering a lattice with real entries.

Theorem 2. *Given rational numbers $\alpha_1, \ldots, \alpha_n$ and $0 < \epsilon < 1$, there is an algorithm that can compute in polynomial time the integers $p_1, \ldots, p_n$ and $q \in \mathbb{Z}^+$ such that*

$$\max_i |q\alpha_i - p_i| < \epsilon \quad \textit{and} \quad q \leq 2^{n(n-3)/4} \cdot 3^n \cdot \epsilon^{-n}.$$

Please refer to the proof in [7].

2.2 Useful lemmas

We start by presenting a crucial lemma regarding the bound of the primes p and q.

Lemma 1. *Let $N = pq$ with $q < p < 2q$. Then*

$$2^{-1/2}N^{1/2} < q < N^{1/2} < p < 2^{1/2}N^{1/2}.$$

Please refer to the proof in [6].

Next, the following lemma shows the bound of $N^2 - \psi(N)$.

Lemma 2. *Let $N = pq$ and $\psi(N) = (p^2-1)(q^2-1)$ where $q < p < 2q$ are large primes. Then*

$$2N - 1 < N^2 - \psi(N) < \frac{5}{2}N - 1.$$

Proof. Suppose $N = pq$ with $q < p < 2q$. We have $N^2 - \psi(N) = p^2 + q^2 - 1 = p^2 + \left(\frac{N^2}{p^2}\right) - 1$. The function f is defined as $f(p) = p^2 + \frac{N^2}{p^2} - 1$. Thus, the derivative $f'(p) = \frac{2p^5 - 2N^2}{p^3}$ which shows that the function f is strictly increasing for the interval p, that is $(N^{1/2}, 2^{1/2}N^{1/2})$. Hence, for $f\left(N^{1/2}\right) < f(p) < f\left(2^{1/2}N^{1/2}\right)$ we have

$$\left(N^{1/2}\right)^2 + \frac{N^2}{\left(N^{1/2}\right)^2} - 1 < p^2 + \frac{N^2}{p^2} - 1 < \left(2^{1/2}N^{1/2}\right)^2 + \frac{N^2}{\left(2^{1/2}N^{1/2}\right)^2} - 1$$

From the left hand side of the inequality, we have $p^2 + \frac{N^2}{p^2} - 1 > 2N - 1$ while for the right hand side of the inequality, we have $p^2 + \frac{N^2}{p^2} - 1 < \frac{5}{2} - 1$. Since $p^2 + \frac{N^2}{p^2} - 1 = N^2 - \psi(N)$, this terminates the proof. □

3 Attack on Equation $e_i x - \psi(N_i)y_i = 1$

We present the first attack in the following theorem.

Theorem 3. *Suppose $k \geq 2, N_i = p_i q_i$ for $1 \leq i \leq k$ be k moduli each with the same size N where $N = \min\{N_i\}$. Let e_i for $i, \ldots, k$ be k public exponents. Define $\delta = \frac{k}{k+1}$. If there exists integer $x < N^\delta$ and k integers $y_i < N^\delta$ such that $e_i x - \psi(N_i)y_i = 1$, then the modulus N is possible to be factored in polynomial time.*

Proof. Suppose $k \geq 2$ and $i = 1, \ldots, k$, then the equation $e_i x - \psi(N_i)y_i = 1$ can be rewritten as

$$e_i x - N_i^2 y_i + N_i^2 y_i - \psi(N_i)y_i = 1$$
$$e_i x - N_i y_i = 1 + y_i(\psi(N_i) - N_i^2)).$$

Dividing by N_i we would have,

$$\begin{aligned}\left|\frac{e_i}{N_i^2}x - y_i\right| &= \frac{\left|1 + y_i(\psi(N_i) - N_i^2\right|}{N_i^2} \\ &< \frac{\left|y_i(1 + \psi(N_i) - N_i^2)\right|}{N_i^2}.\end{aligned}$$

From Lemma 2, $N^2 - \psi(N) < \frac{5}{2} + 1$ and taking $N = \min\{N_i\}$ hence,

$$\begin{aligned}\frac{\left|y_i(1 + \psi(N_i) - N_i^2)\right|}{N_i^2} &< \frac{N^\delta\left(\frac{5}{2}N + 2\right)}{N_i^2} \\ &< \frac{N^\delta\left(\frac{5}{2}N\right)}{N^2} \\ &= \frac{5}{2}N^{\delta-1}.\end{aligned}$$

Thus we have

$$\left|\frac{e_i}{N_i}x - y_i\right| < \frac{5}{2}N^{\delta-1}. \tag{1}$$

It can be seen that the inequality in 1 relates to the condition stated in Theorem 2 which is $|q\alpha_i - p_i| < \epsilon$.

Now, we proceed to show the existence of the integer x and the integers y_i. Assuming $\epsilon = \frac{5}{2}N^{\delta-1}$ and $\delta = \frac{k}{(k+1)}$, we have

$$\begin{aligned}N^\delta \cdot \epsilon^k &= N^\delta\left(\frac{5}{2}N^{\delta-1}\right)^k \\ &= \left(\frac{5}{2}\right)^k.\end{aligned}$$

Then, since $\left(\frac{5}{2}\right)^k = (2.5)^k < 2^{\frac{k(k-3)}{4}} \cdot 3^k$ for $k \geq 2$, we get $N^\delta\epsilon^k < 2^{\frac{k(k-3)}{4}} \cdot 3^k$. Thus, if $d < N^\delta$ then, $x < 2^{\frac{k(k-3)}{4}} \cdot 3^k \cdot \epsilon^{-m}$. In summary,

$$\left|\frac{e_i}{N_i}x - y_i\right| < \epsilon, \quad x < 2^{\frac{k(k-3)}{4}} \cdot 3^k \cdot \epsilon^{-m}.$$

which requires the condition of Theorem 2 to be fulfilled in order to find both integers x and y_i for $i = 0, \ldots, k$. Using these values, one can compute

$$\begin{aligned}\psi(N_i) &= \frac{e_i x - 1}{y_i} \\ &= (p_i^2 - 1)(q_i^2 - 1).\end{aligned} \tag{2}$$

Expanding and rearrange Eq. (2), we have

$$p_i^2 + q_i^2 = N_i^2 - \psi(N_i) + 1. \tag{3}$$

Also, we have the information,

$$\begin{aligned}(p_i + q_i)^2 &= p_i^2 + q_i^2 + 2N_i \\ &= N_i^2 - \psi(N_i) + 1 + 2N_i \\ p_i + q_i &= \sqrt{(N_i^2 - \psi(N_i) + 1 + 2N_i)}.\end{aligned} \tag{4}$$

Substituting the value from Eq. (4) into quadratic equation,

$$\begin{aligned} X^2 - (p_i + q_i)X + N_i &= 0 \\ X^2 - (p_i + q_i)X + N_i &= 0 \\ (X - p_i)(X - q_i) &= 0. \end{aligned} \tag{5}$$

Solving Eq. (5) would give us roots of p_i and q_i. □

The following algorithm summarizes the steps of factoring the modulus N.

Algorithm 3.1 Factoring k RSA moduli simultaneously via Theorem 3

Input: The public RSA key pairs (N_i, e_i) for $i = 1, \ldots, k$.
Output: The primes factors p_i and q_i or $\perp$.

1. Set $N = \min N_1, \ldots, N_k$.
2. Compute $\delta = \frac{k}{k+1}$.
3. Compute $\varepsilon = \frac{5}{2} N^{\delta - 1}$.
4. Compute $C = 3^{k+1} \cdot 2^{\frac{(k+1)(k-4)}{4}} \cdot \varepsilon^{-k-1}$.
5. Compute lattice $\mathcal{L}$ spanned by the rows of the matrix $\mathcal{M}$ as shown in the proof of Theorem 2.
6. Compute matrix $\mathcal{K}$ by applying LLL algorithm onto $\mathcal{M}$.
7. Compute matrix $\mathcal{H} = \mathcal{K}\mathcal{M}^{-1}$.
8. Assign each element in the first row of $\mathcal{H}$ (starting from the most left) as $X, Y_1, \ldots, Y_k$ respectively.
9. **for** $i = 2, 3, \ldots, k$ **do**
10. Compute $\psi(N_i) = \frac{e_i X - 1}{Y_i}$.
11. Let $S_i = p_i + q_i$. Compute $S_i = \sqrt{N_i^2 - \psi(N_i) - 1 - 2N_i}$.
12. Compute $p_i, q_i = X^2 - S_i X + N_i = 0$.
13. **if** $p_i, q_i \in \mathbb{Z}$ then output p_i, q_i.
14. **else** Algorithm fails or $\perp$.
    end if
    end for

3.1 Examples

The following example illustrates Algorithm 3.1.

Example 1. Consider the three pairs of the RSA moduli and its corresponding public exponents.

$N_1 = 25322076580483238005226202489735088141 1,$
$e_1 = 83524264742194137817506120546690891778368342106398561944680742906646552111 47,$
$N_2 = 19490894461013320019366060166789533574 7,$
$e_2 = 21345516244339126547661522845471933825221413676280512067343917970910338062987,$
$N_3 = 12748677229376767602299230716191668965 7,$
$e_3 = 90413810519149777997306656750128489363486581395698336007780209331 4641425227.$

Then, we set

$$N = \min(N_1, N_2, N_3) = 12748677229376767602299230716191668965 7$$

For $k = 3$, we obtain $\frac{k}{k+1} = 0.375$ and $\varepsilon = \frac{5}{2} \cdot N^{\delta - 1} \approx$. Then, applying Theorem 2, we compute $C = [3^{k+1} \cdot 2^{(k+1)(k+4)4} \cdot \varepsilon^{-k-1}] = 13217828551417832650063842406547522383 6$

We also compute $C_i = [-\frac{C \cdot e_i}{N_i^2}]$ for $i = 1, 2, 3$ and obtain

$$\begin{aligned}
C_1 &= -172176604905140834530411480550053951 62, \\
C_2 &= -742682579789499500236664749508943272 91, \\
C_3 &= -735300118276169726080959904848588310 74
\end{aligned}$$

Then, we compute a lattice $\mathcal{L}$ spanned by the rows of the matrix

$$\mathcal{M} = \begin{bmatrix} 1 & C_1 & C_2 & C_3 \\ 0 & C & 0 & 0 \\ 0 & 0 & C & 0 \\ 0 & 0 & 0 & C \end{bmatrix}$$

Performing the LLL algorithm to the Matrix M would yield

$$\mathcal{K} = \begin{bmatrix} \mathcal{K}_{11} & \cdots & \mathcal{K}_{14} \\ \vdots & \ddots & \vdots \\ \mathcal{K}_{41} & \dots & \mathcal{K}_{44} \end{bmatrix}$$

$$\begin{aligned}
\mathcal{K}_{11} &= 255929320328483 \\
\mathcal{K}_{12} &= -91972327270006 \\
\mathcal{K}_{13} &= 112017302482815 \\
\mathcal{K}_{14} &= 342005142641862 \\
\mathcal{K}_{21} &= -663502308951713181165929548783738 \\
\mathcal{K}_{22} &= -134436512625207424992765164471306 4 \\
\mathcal{K}_{23} &= 247898884687663360659119181180 90 \\
\mathcal{K}_{24} &= 126864785816002898843323125868456 \\
\mathcal{K}_{31} &= 114340545404696490585886987123178 5 \\
\mathcal{K}_{32} &= 836705222972193405653038907009 98 \\
\mathcal{K}_{33} &= -109936465778645957719069813628142 7 \\
\mathcal{K}_{34} &= -47305646749816652547300226858063 4 \\
\mathcal{K}_{41} &= -139197228147773281154567121162232 0 \\
\mathcal{K}_{42} &= 618514744958873746139420052592244 \\
\mathcal{K}_{43} &= -940533386714458854516613095675808 \\
\mathcal{K}_{44} &= 151602624840379191447200733279551 6
\end{aligned}$$

Next, computing matrix $\mathcal{H} = \mathcal{K} \cdot \mathcal{M}^{-1}$ gives us

$$\mathcal{H} = \begin{bmatrix} \mathcal{H}_{11} & \cdots & \mathcal{H}_{14} \\ \vdots & \ddots & \vdots \\ \mathcal{H}_{41} & \dots & \mathcal{H}_{44} \end{bmatrix}$$

where

$$\begin{aligned}
\mathcal{H}_{11} &= 255929320328483\\
\mathcal{H}_{12} &= 33337579844090\\
\mathcal{H}_{13} &= 143801417249388\\
\mathcal{H}_{14} &= 142371993081789\\
\mathcal{H}_{21} &= -663502308951713181165929548783738\\
\mathcal{H}_{22} &= -864283981726890448056230994810 45\\
\mathcal{H}_{23} &= -372808290402351978264360031846563\\
\mathcal{H}_{24} &= -369102477272164235313182555339621\\
\mathcal{H}_{31} &= 114340545404696490585869871231785\\
\mathcal{H}_{32} &= 148941006718316147817481800779188\\
\mathcal{H}_{33} &= 642455989691207191594557284517128\\
\mathcal{H}_{34} &= 636069807024518251790756385697546\\
\mathcal{H}_{41} &= -139197228147773281154567121162232 0\\
\mathcal{H}_{42} &= -181319541719423356400564314563661\\
\mathcal{H}_{43} &= -782120573724998382998562521131348\\
\mathcal{H}_{44} &= -774346088108350775795979616399999
\end{aligned}$$

Observe from the first row of the matrix $\mathcal{H}$,

$$\begin{aligned}
X &= 255929320328483,\\
Y_1 &= 33337579844090,\\
Y_2 &= 143801417249388,\\
Y_3 &= 142371993081789.
\end{aligned}$$

Next, we compute $\psi(N_i) = \frac{e_i X - 1}{Y_i}$,

$\psi(N_1) = 641207562347857675949642198305012130943652959941203072090089020919731404800 00$
$\psi(N_2) = 379894966890359718704364887095457624421522486554907329152205202372044755814 40$
$\psi(N_3) = 162528771098829695545256215177602494765356552485348306566037377486201592297 60$

Since we know$\psi(N_i)$, we can compute $S_i = \sqrt{N_i^2 - \psi(N_i) + 1 + 2N_i}$,

$$\begin{aligned}
S_1 &= 31877810089252638588\\
S_2 &= 28373881244619319492\\
S_3 &= 22588540125047578198
\end{aligned}$$

Utilising values of S_i, we can solve for p_i, q_i from the quadratic equation $X^2 - S_i X + N_i = 0$. Finally we have

$$\begin{aligned}
p_1 &= 15028999435905310589\\
p_2 &= 16708911996216745859\\
p_3 &= 11565865260296943587\\
q_1 &= 16848810653347327999\\
q_2 &= 11664969248402573633\\
q_3 &= 11022674864750634611
\end{aligned}$$

4 Conclusion

In conclusion, our contribution to this study is factoring the modulus of RSA given a Diophantine equation $e_i x - \psi(N_i) y_i = 1$ where every variable but x has distinct values. We showed that it is feasible to recover the private exponents p_i, q_i via lattice basis reduction if certain conditions are fulfilled.

5 Acknowledgement

We would like to express our gratitude to the anonymous reviewers for the invaluable insights to improve the quality of our manuscript.

References

[1] Bunder, M., Nitaj, A., Susilo, W., and Tonien, J. : A generalized attack on RSA type cryptosystems, Theoretical Computational Science, vol. 704, pp. 74-81, 2017.

[2] Bunder, M., Nitaj, A., Susilo, W., and Tonien, J. : Cryptanalysis of RSA-type cryptosystems based on Lucas sequences, Gaussian integers and elliptic curves, Journal of Information Security and Applications, vol. 40, pp.193-198, 2018.

[3] Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$, Advances in Cryptology-Eurocrypt'99, Lecture Notes in Computer Science Vol. 1592, Springer-Verlag, pp. 1-11, 1999.

[4] Kuwakado, H., Koyama, K., and Tsuruoka, Y.: A new RSA-type scheme based on singular cubic curves $y^2 = x^3 + bx^2 (\mod n)$, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E78-A, 27-33, 1995.

[5] Lenstra, A.K., Lenstra, H.W., and Lovász, L.: Factoring polynomials with rational coefficients, Mathematische Annalen, Vol. 261, pp. 513-534, 1982.

[6] Nitaj, A. New weak RSA keys, JP Journal of Algebra, Number Theory and Applications, Vol.23(2), 131-148, 2011.

[7] Nitaj, A., Ariffin, M.R.K., Bahig, and D.I. H.M.: New attacks on the RSA cryptosystem, in D. Pointcheval and D. Vergnaud (Eds.): AFRICACRYPT 2014, LNCS 8469, Springer, pp. 178-198, 2014.

[8] Rahman, M.N.A., Arifiin, M.R.K., Asbullah, M.A., Yunos, F. New Vulnerability on System of $N_i = p_i^2 q_i$ Using Good Approximation of $\phi(N)$, Proceedings of the 6th International Cryptology and Information Security Conference 2018, pp. 139-150, 2018.

[9] Rivest, R., Shamir, A., and Adleman, L. A Method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol. 21 (2), pp. 120-126, 1978.

[10] Ruzai, W.N.A., Ariffin, M.R.K., Asbullah, M.A., Ghafar, A.H.A., New simultaneous Diophantine Attacks on Generalized RSA Key Equations, Journal of King Saud University-Computer and Information Sciences, Elsevier, Vol. 36(5), pp. 102074, 2024.

[11] Wiener, M.: Cryptanalysis of short RSA secret exponents, IEEE Transactions on Information Theory, Vol. 36, pp. 553-558, 1990.

[12] Zheng, M., Kunihiro, N., and Hu, H.: Cryptanalysis of RSA Variants with Modified Euler Quotient. In: Joux A., Nitaj A., Rachidi T. (eds) Progress in Cryptology-AFRICACRYPT 2018. LNCS 10831. Springer, 2018.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 149–162.

Modification of Stickel's Key Exchange Scheme Using Matrix Power Function Over Tropical Semiring

Ahmad Rufa'i[1,2] and Hailiza Kamarulhaili[1]

[1] School of Mathematical Sciences, Universiti Sains Malaysia, 11800 Minden, Pulau Pinang
ahmadrufai35@student.usm.my,hailiza@usm.my
[2] Department of Mathematics, Sokoto State University, Sokoto, Nigeria
ahmad.rufai@ssu.edu.ng

Abstract. The Diffie-Hellman key exchange scheme was the first public key protocol invented for exchanging cryptographic keys securely over a public channel. Stickel's key exchange scheme is a revolution of the Diffie-Hellman scheme over a non-commutative structure. Stickel's scheme and the tropical form of the scheme are shown to be vulnerable to linear attack and display of patterns by higher powers of tropical matrices, respectively. In this work, we improve the security of Stickel's key exchange scheme. The improved scheme employed matrix power function (MPF) over tropical semiring. The tropical semiring protects the improved scheme from linear attack because tropical matrices are generically non-invertible. Moreover, the matrix power function replaced the power of tropical matrices that displayed the number of operations carried out to obtain the public key by a matrix power. The improved scheme is expected to be more secure than Stickel's key exchange scheme and the tropical form of the scheme.

Keywords: public-key cryptography · Stickel's key exchange Scheme · tropical semiring · matrix power function

1 Introduction

Cryptography plays a crucial role in today's digital world, guaranteeing the confidentiality and integrity of digital transactions. Public key cryptography lies at the core of the task; it allows two parties to interact privately even in the absence of pre-arranged security keys. Group theory provides a diverse cryptography challenge, especially regarding non-abelian groups [11]. Construction of Public key cryptography on non-abelian groups is a relatively recent field of study. Stickel key exchange scheme [20] lies in non-abelian group-based cryptography, the author used matrices in a group of $n \times n$ invertible matrices as a platform. The scheme was attacked due to the invertibility of the chosen platform (linear attack). Grigoriev and Shpilrain [8, 9] improved the scheme using tropical semiring as a platform to make it resistant to linear attack. The tropical schemes were shown to be vulnerable to attacks by Kotov and Ushakov [12], Rudy and Monico [16] and other related attacks. The attacks exploited the periodicity of tropical matrices. The display of patterns by the higher power of matrices is the main drawback of tropical algebra in cryptography [1].

In this paper, we propose an improved key exchange scheme. The proposed scheme is an improvement of Stickel's key exchange scheme. The improved scheme uses the exponent variant of Stickel's scheme rather than the polynomial variant that was used by Grigoriev and Shpilrain [8] in their tropical version of Stickel's scheme. The advantage of tropical protocols over classical protocols is improved efficiency because when conducting matrix

multiplication in a tropical sense one need not perform any multiplication. The other advantage is tropical matrices are generically not invertible which makes the matrices less vulnerable to linear attack. The employment of matrix power function over tropical semiring in the improved scheme reduces the pattern displayed by the higher powers of tropical matrices.

The remaining part of the paper is organized as follows: section 2 gives basic definitions and terms on tropical algebra and some of the operations defined on it. Section 3 discusses some of the related works on matrix algebra and tropical matrix algebra. Section 4 presents a proposed key exchange scheme, security analysis and an example. Section 5 gives a performance comparison between the existing schemes and the proposed scheme. Section 6 showcases some properties of tropical matrices in terms of propositions and corollaries. Finally, The conclusion is presented in section 6.

2 Definitions of Basic Terms

Definition 1. [7] **Ring:** a non-empty set R with two binary operations $+$ and $\times$ denoted by $(R, +, \times)$ is called a ring if the following properties hold:

1. $(R, +)$ is an abelian group.
2. $(R, \times)$ is a semi-group.
3. Distributivity of $\times$ over $+$.

Definition 2. [6] **Semiring:** a non-empty set R with two binary operations $+$ and $\times$ is called a semiring if it satisfies the following axioms:

1. $((R, +)$ is a commutative monoid with identity element 0
2. $((R, \times)$ is a monoid with identity element 1
3. Multiplication distributes over addition.
4. $0a = a0 = 0$ for all a .

Definition 3. [13] **Tropical Semiring:** Let $\mathbb{R}$ be the set of real numbers, the set $\mathbb{R}_{min} = \mathbb{R} \cup \infty$ (the extended set of real numbers) with two binary operations $\oplus$ and $\otimes$ defined by $a \oplus b = \min(a, b)$ and $a \otimes b = a + b$ is called a tropical semi-ring if it is closed under $\oplus$ and it contains 0 and ∞. The set $R_{max} = \mathbb{R} \cup -\infty$ with two binary operations $\oplus$ and $\otimes$ defined by $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ is also a tropical semi-ring.

The structures satisfy the following properties:

1. Commutativity: for all a, b in $\mathbb{R}_{min}$ and $\mathbb{R}_{max}$
   $a \oplus b = b \oplus a$
   $a \otimes b = b \otimes a$
2. associativity: for all a, b, c in $\mathbb{R}_{min}$ and $\mathbb{R}_{max}$ we have
   $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
   $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
3. Distributivity: for all a, b, c in $\mathbb{R}_{min}$ and $\mathbb{R}_{max}$
   $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$

4. Idempotency: $a \oplus a = a$ for all a in $\mathbb{R}_{min}$ and $\mathbb{R}_{max}$.

Definition 4. [17] **Matrix power function:** Let a and B be two matrices. The left-sided matrix power function of the power matrix B and base matrix a is defined by ${}^{B}a = X = x_{ij} = \prod_{k=1}^{m} a_{jk}^{b_{ik}}$. The right-sided matrix power function $a^{B} = Y = y_{ij} = \prod_{k=1}^{m} a_{ik}^{b_{kj}}$ and the two-sided matrix power function of power matrices B and C and base matrix a is given by ${}^{B}a^{C} = Z = z_{ij} = \prod_{k=1}^{n}\prod_{l=1}^{n} a_{kl}^{b_{ik}.y_{lj}}$. Furthermore, given

$$a = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

then,

$${}^{B}a = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} a_{11}^{b_{11}} a_{21}^{b_{12}} & a_{12}^{b_{11}} a_{22}^{b_{12}} \\ a_{11}^{b_{21}} a_{21}^{b_{22}} & a_{11}^{b_{21}} a_{22}^{b_{22}} \end{bmatrix}$$

and

$$a^{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}^{b_{11}} a_{12}^{b_{21}} & a_{11}^{b_{12}} a_{12}^{b_{22}} \\ a_{21}^{b_{11}} a_{22}^{b_{21}} & a_{21}^{b_{12}} a_{22}^{b_{22}} \end{bmatrix}.$$

Definition 5. [2] **Circulant Matrix:** This is a square matrix $n \times n$, where the $i^{t}h$ row of the matrix is obtained by right shifting $(i-1)^{th}$ row cyclically by one position, the general form of a circulant matrix is given below:
.

$$C = \begin{bmatrix} c_1 & c_2 \dots & c_n \\ c_n & c_1 \dots & c_{n-1} \\ \vdots & \vdots & \vdots \\ c_2 & c_3 \dots & c_1 \end{bmatrix}.$$

Definition 6. [4] **Tropical Matrix algrbra:** The subset of the tropical semiring $M = \mathbb{Z} \cup [\infty]$, the set of $k \times k$ matrices can be furnished with two binary operations $\oplus$ and $\otimes$ such that $(a_{ij}) \oplus (b_{ij}) = (a_{ij} \oplus b_{ij})$ and $(a_{ij}) \otimes (b_{ij}) = (a_{1i} \otimes b_{1j} \oplus \ldots \oplus a_{ni} \otimes b_{nj})$, the algebra $M_k(\mathbb{Z}_{min})$ is called tropical matrix algebra.

Definition 7. [12] **Tropical Polynomial:** Let $x_1, x_2, \ldots, x_n$ be the elements in $(S, \oplus, \otimes) \subset \mathbb{Z}_{min}$. The product of the elements is called a monomial. Tropical Polynomial is a linear combination of tropical monomial. Generally, tropical polynomial is given by $p(x) = \oplus_{i=0}^{n}(a_i \otimes x^{i})$.

Definition 8. [15] **Linear Periodic:** a sequence of matrices $\{a^{n}, n \in \mathbb{N}\}$ is called periodic if there exist a period ρ, a linear factor ϵ and some defect d such that for $p > d$, the following equation holds:

$$a_{ij}^{n+\rho} = \epsilon + a_{ij}^{n}.$$

3 Related Works

In 2005, key exchange schemes were proposed by Stickel in [20], which are generalizations of discrete logarithm problem (DLP) over non-abelian groups. Let G be a non-abelian group and the elements $a, b \in G$ do not commute, that is $ab \neq ba$. Let $n_1 = |a|$ and $n_2 = |b|$. For key transmission, two random numbers would be chosen by the first party (a) and a second party (B) such that $(r, v) < n_1, (s, w) < n_2$ and $r, s, v, w \in N$.

Table 1: algorithm 1: Stickel's Key-Exchange Scheme (2005)

Input	$n \times n$ matrices a, b as public and natural numbers r, v, s, w as private
Steps	
1.	Party (a) Chooses a, b and r, s and computes $u = a^r b^s$ then send u to party (B)
2.	Party (B) Chooses a, b and v, w and computes $v = a^v b^w$ then send v to party (a)
3.	Party (a) computes $K_a = a^r v b^s$
4.	Party (B) computes $K_B = a^v u b^w$
Output	Shared secret key $K_a = K_B$

The author suggested a more general scheme and a polynomial scheme, which are like the above system. The schemes were cryptanalyzed by Shpilrain [19] in 2008. The choice of the group of invertible matrices $GL(n, F)$ on a finite field F makes the schemes prone to linear attack. It is enough for an attacker to find matrices x and y such that

$$\begin{aligned} xa &= ax \\ yb &= by \\ xy &= u \end{aligned} \tag{2}$$

This can be translated into a system of $3k^2$ linear equations with $2k^2$ unknowns, where a, b, u are known; x, y are unknown and k is the size of the matrix. For the general scheme, the third equation in the above system is non-linear i.e.,$u = xwy$.The entire system can be rewritten as

$$\begin{aligned} awyu^{-1} &= wyu^{-1}a \\ by &= yb \end{aligned} \tag{3}$$

With k^2 unknowns and $2k^2$ linear equations. If $k = 2$, equation (2) will have 12 equations with 8 unknowns and equation (3) will have 4 equations with 8 unknowns. If the solutions of the equation (2) and (3) are obtained an attacker can use v which is one of the published keys to obtain the secret key. For example, in algorithm (1) $xvy = xa^r b^w y = a^v xyb^w = a^v ub^w = K_B$

.

In 2014, Grigoriev and Shpilrain [8] proposed tropical algebra as a platform to improve one of the Stickel's schemes (polynomial version). Let $S = (M_n\mathbb{Z}, \oplus, \otimes)$ be a set of tropical matrices over $\mathbb{Z}$, the set of integers. Let $a, b \in S$ such that $a \otimes b \neq b \otimes a$.

Table 2: algorithms 2:Tropical Key-Exchange Scheme (Shpilrain, 2014)

Input	$a, b \in M_n(\mathbb{Z}, \oplus, \otimes)$ and polynomials $p_1(x)$,$p_2(x)$ and $q_1(x)$,$q_2(x)$ as private
Steps	
1.	alice chooses randomly two polynomials $p_1(x)$,$\mathrm{p}_2(x)$ and sends $u = p_1(a) \otimes p_2(b)$ to Bob
2.	Bob randomly selects $q_1(x)$,$q_2(x)$ and sends $v = q_1(a) \otimes q_2(b)$ to alice.
3.	alice computes $K_a = p_1(\mathrm{a}) \otimes v \otimes p_2(b)$
4.	Bob computes $K_a = q_1(\mathrm{a}) \otimes u \otimes q_2(b)$
Output	Shared secret key $K_a = K_B$

The tropical version of the scheme was attacked by Kotov and Oshakov [12] in 2018 due to the parameter selection of the scheme and the tropical polynomial used in the scheme. They presented a general attack by finding $X = \otimes_{i=0}^{D} x_i \otimes a^{\otimes i}$ and $Y = \otimes_{j=0}^{D} y_j \otimes B^{\otimes j}$ such that $(\oplus_{i=0}^{D} x_i \otimes a^{\otimes i}) \otimes (\oplus_{j=0}^{D} y_j \otimes B^{\otimes j}) = U$ where x_i and y_j are integers. This can be written as $\oplus_{i,j=0}^{D} x_i \otimes y_j \otimes V^{ij} = U$, where $V^{ij} = a^{\otimes i} \otimes B^{\otimes j}$. The solution of $\min_{i,j}(x_i + y_j, T_{kl}^{ij}) = 0$ is enough to break the scheme, where $k, l \in [1, n]$, $T_{kl}^{ij} = V_{kl}^{ij} - U_{kl}, (x_i + y_j)$ is the unknown, T_{kl}^{ij} is the coefficient, and kl, ij play a role of rows and columns respectively. In 2019, Grigoriev and Shpilrain [9] developed a tropical key exchange protocol based on the semi-direct product of semigroup purposely to avoid the homogeneous patterns produced by the higher power of matrices. Subsequently, Rudy and Monico [16] presented an attack on this protocol. The basis of the attack solely relies on the property of the sequence $(a)^{(k)}k \geq 1$ where $a^{(k)} = (M, H)^k$ is linearly ordered in the max-plus algebra, M and H are public matrices used in the scheme. The property can give way to an attacker to use a binary search method for finding the secret keys [16]. In the same year (2020), Isaac and Kahrobaei [10] came up with an attack using a different approach. They assume that the sequence formed in the cryptographic processes of the scheme is ultimately periodic. The assumption implies that $a_{ij}^{k+p} = \epsilon + a_{ij}^{k}$ for all $k \geq D$, where D is the bigger number than the maximal degree of any tropical polynomial that could be used in the scheme. The assumption holds when H is irreducible. The attack is more efficient in practice [10]. In 2022, Muanalifah and Sergeev [14] used a tropical discrete logarithm to suggest an attack on the tropical semidirect product.

4 Proposed Key Exchange Scheme

In this paper, an improved key exchange protocol has been proposed using matrix power function (MPF) problem, that is, given a base matrix a and matrix power function M find a matrix Y such that $M = a^{\otimes Y}$, and a semiring action problem, that is, given a matrix K and a matrix $T \in U \otimes K \otimes V$, find two matrices U and V such that $T = U \otimes K \otimes V$. If

$$a = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

then,MPF on tropical matrix algebra is defined by

$$a^{\otimes B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{\otimes} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}^{\otimes b_{11}} \otimes a_{12}^{\otimes b_{21}} & a_{11}^{\otimes b_{12}} \otimes a_{12}^{\otimes b_{22}} \\ a_{21}^{\otimes b_{11}} \otimes a_{22}^{\otimes b_{21}} & a_{21}^{\otimes b_{12}} \otimes a_{22}^{\otimes b_{22}} \end{bmatrix}$$

Let S be a tropical semiring, $X_1, X_2, Y_1, Y_2, Z_1, Z_2, Q, R \in CM_k(\mathbb{Z}_{min}) \subset S$ be a tropical circulant matrices in S and V be a matrix in $M_k(\mathbb{Z}_{min})$. The following steps will be followed by Salima and Salim to share a secret key between them after agreeing on $X_1, X_2 \in CM_k(\mathbb{Z}_{min})$ and $V \in M_k(\mathbb{Z}_{min})$.

Table 3: algorithm 3: Proposed Key Exchange Scheme

Input	$X_1, X_2 \in CM_k(\mathbb{Z}_{min})$,V$\in M_k(\mathbb{Z}_{min})$ and $Y_1, Y_2 \in CM_k(\mathbb{Z}_{min}), Q, R \in CM_k(\mathbb{Z}_{min})$
Steps	
1.	Salima choose$Y_1, Y_2, Q \in CM_k(\mathbb{Z}_{min})$ as private keys compute: $a = Q \otimes (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2})$ $B = a \otimes V$, she then sends B to Salim.
2.	Salim chooses $Z_1, Z_2, R \in CM_k(\mathbb{Z}_{min})$ and compute $C = R \otimes (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2})$ $D = C \otimes V$, he then sends D to Salima.
3.	Salima computes $K_a = a \otimes D$
4.	Salim computes $K_B = C \otimes B$
Output	Shared secret key $K_a = K_B$

4.1 Correctness of the scheme

Proposition 1. *The two parties (Salima and Salim) shared the same key in algorithm (3).*

Proof. To show that Salima and Salim shared the same key, we need to show that

$$K_a = K_B$$

From the proposed algorithm we have

$$\begin{aligned} \mathrm{K}_a &= a \otimes D \\ &= \mathrm{a} \otimes C \otimes V \\ &= [\mathrm{Q} \otimes (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2})] \otimes [R \otimes (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2})] \otimes V \\ &= [\mathrm{Q} \otimes R] \otimes [(X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2}) \otimes (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2})] \otimes V \\ &= [\mathrm{Q} \otimes R][(X_1^{\otimes Y_1} \otimes X_1^{\otimes Z_1} \otimes X_2^{\otimes Y_2} \otimes X_2^{\otimes Z_2}] \otimes V \end{aligned}$$

Similarly,

$$\begin{aligned} \mathrm{K}_B &= C \otimes B \\ &= \mathrm{C} \otimes a \otimes V \\ &= [\mathrm{R} \otimes (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2})] \otimes [Q \otimes (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2})] \otimes V \\ &= [\mathrm{R} \otimes Q] \otimes [(X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2}) \otimes (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2})] \otimes V \\ &= [\mathrm{R} \otimes Q][(X_1^{\otimes Z_1} \otimes X_1^{\otimes Y_1} \otimes X_2^{\otimes Z_2} \otimes X_2^{\otimes Y_2}] \otimes V \\ &= [\mathrm{Q} \otimes R] \otimes [X_1^{\otimes Y_1} \otimes X_1^{\otimes Z_1} \otimes X_2^{\otimes Y_2} \otimes X_2^{\otimes Z_2}] \otimes V \end{aligned}$$

Since $(CM_k(\mathbb{Z}_{min}, \oplus, \otimes)$ is commutative. Thus,

$$\mathrm{K}_a = K_B = [Q \otimes R] \otimes [X_1^{\otimes Y_1} \otimes X_1^{\otimes Z_1} \otimes X_2^{\otimes Y_2} \otimes X_2^{\otimes Z_2}] \otimes V$$

This shows that the two parties shared the same key.

□

Example 1. Toy example of the proposed key-exchange scheme

Let $\mathrm{X}_1 = \begin{bmatrix} 7 & 1 & 5 \\ 5 & 7 & 1 \\ 1 & 5 & 7 \end{bmatrix}, X_2 = \begin{bmatrix} 6 & 4 & 3 \\ 3 & 6 & 4 \\ 4 & 3 & 6 \end{bmatrix} \in CM_k(\mathbb{Z}_{min}, \oplus, \otimes)$

Salima chooses

$$\mathrm{Y}_1 = \begin{bmatrix} 5 & 6 & 7 \\ 7 & 5 & 6 \\ 6 & 7 & 5 \end{bmatrix}, Y_2 = \begin{bmatrix} 11 & 8 & 3 \\ 3 & 11 & 8 \\ 8 & 3 & 11 \end{bmatrix},$$

$$\mathrm{Q} = \begin{bmatrix} 9 & 14 & 19 \\ 19 & 9 & 14 \\ 14 & 19 & 9 \end{bmatrix} \in CM_3(\mathbb{Z}_{min}, \oplus, \otimes)$$

as private keys and computes:

$$a = Q \otimes (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2})$$

$$= \begin{bmatrix} 9 & 14 & 19 \\ 19 & 9 & 14 \\ 14 & 19 & 9 \end{bmatrix} \otimes \begin{bmatrix} 165 & 163 & 155 \\ 155 & 165 & 163 \\ 163 & 155 & 165 \end{bmatrix}$$

$$= \begin{bmatrix} 169 & 172 & 164 \\ 164 & 169 & 172 \\ 172 & 164 & 169 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} 169 & 172 & 164 \\ 164 & 169 & 172 \\ 172 & 164 & 169 \end{bmatrix} \otimes \begin{bmatrix} 2 & 10 & 21 \\ 5 & 17 & 3 \\ 7 & 8 & 14 \end{bmatrix}$$

$$= \begin{bmatrix} 171 & 172 & 175 \\ 166 & 174 & 172 \\ 169 & 177 & 167 \end{bmatrix}$$

Salim chooses $Z_1, Z_2, R \in CM_3(\mathbb{Z}_{min}, \oplus, \otimes)$ as private keys and compute:

$$C = R \otimes (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2}) \text{ and } D = C \otimes V$$

$$\mathrm{C} = \begin{bmatrix} 94 & 93 & 95 \\ 95 & 94 & 93 \\ 93 & 95 & 94 \end{bmatrix}$$

$$\mathrm{D} = \begin{bmatrix} 96 & 103 & 96 \\ 97 & 101 & 97 \\ 95 & 102 & 98 \end{bmatrix}$$

Salima computes $K_a = a \otimes D = \begin{bmatrix} 259 & 266 & 262 \\ 260 & 267 & 260 \\ 261 & 265 & 261 \end{bmatrix}$

Salim computes $K_B = C \otimes B = \begin{bmatrix} 259 & 266 & 262 \\ 260 & 267 & 260 \\ 261 & 265 & 261 \end{bmatrix}$

4.2 Security analysis of the proposed Scheme

Unlike the original Stickel's Scheme and the original tropical version of the scheme, the power of tropical matrices revealed the number of operations carried out for obtaining the public key, the exponent in the improved scheme is a matrix, not a natural number. Thus, the operations do not easily leak information to the adversary. The theorem below established the periodicity of tropical matrix powers.

Theorem 1. *[3] if $a \in M_k(\mathbf{Z}_{max})$ and a is irreducible then the sequence of matrix $a^{\otimes n}$ (for $n \in \mathbb{N}$) is eventually periodic with period $\gamma(a)$ and ratio $\lambda(a)$.*

Proof. Perron-Frobenius theorem states that if $a \in M_k(\mathbf{Z}_{max})$ and a is irreducible, there exists a unique eigen value $\lambda(a)$ such that

$$a \otimes v = \lambda \otimes v$$

for $k \geq \mathbb{N}$ and some eigenvector v.
The eigenvalue $\lambda(a)$ is related to the cycles in the graph representation of a from maximum cycle mean, thus;

$$\lambda(a) = \max_{\sigma} \mu(\sigma, a) = \frac{\omega(\sigma,a)}{l(\sigma)}$$

where σ denotes a cycle in the graph of a. Thus, the length of the cycles of the graph of a is the period $\gamma(a)$ and the matrix power $a^{\otimes n}$ is the walk of length n in the graph.
considering the properties of the max-plus algebra and irreducible matrices, it can be shown that there exists N such that for all $n \geq N$

$$a^{\otimes(n+\gamma(a))} = \lambda(a)^{\gamma(a)} \otimes a^{\otimes n}$$

Hence, $a^{\otimes n}$ is eventually periodic with period $\gamma(a)$ and ratio $\lambda(a)$ due to the eigenvalue and irreducibility.

□

Sergeev, S. [18] expressed that the theorem can be written in form of CSR expansion, i.e., there exist non-negative integer T such that $\forall_t \geq T : a^{\otimes t} = \lambda^{\otimes t} \otimes CS^tR$ where C, S and R are matrices defined in terms of a.

For example, if X is a tropical matrix, the powers of X displays periodicity for instance, if

$$\text{X}= \begin{bmatrix} 5 & 9 & -2 \\ -5 & -1 & 9 \\ 0 & 2 & -2 \end{bmatrix}$$

is a tropical matrix, each successive element of the power of X from $n = 2$ to $n = 7$ differs by a linear factor $5, 6$ or 7 from the previous element and the pattern repeats after every third power, resulting in $\rho = 3$ and the defect $\delta = 3$. The relation can leak information to an attacker on how to recover the matrix key as shown by [1], but in the proposed scheme if $\text{X}= \begin{bmatrix} 2 & 5 & 4 \\ 3 & 6 & 7 \\ 2 & 7 & 5 \end{bmatrix}$, $\text{N}= \begin{bmatrix} 7 & 3 & 5 \\ 8 & 1 & 3 \\ 9 & 2 & 4 \end{bmatrix}$ we have $X^{\otimes N} = \begin{bmatrix} 99 & 19 & 41 \\ 132 & 29 & 61 \\ 115 & 23 & 51 \end{bmatrix}$ which does not show any pattern of tropical power of matrices that can leak information to an attacker.

Moreover, assuming an attacker possessed the published keys X_1, X_2, and B, he/she is expected to find Y_1, Y_2 or Z_1, Z_2 in $M_k(\mathbb{Z}_{min})$ such that $B = (X_1^{\otimes Y_1} \otimes X_2^{\otimes Y_2}) \otimes V$ or $D = (X_1^{\otimes Z_1} \otimes X_2^{\otimes Z_2}) \otimes V$ to attack the scheme. Similarly, given some matrices, the matrix power function (MPF) is a one-way function and the solution to this type of equation is not feasible from the literature we have reviewed [5].

4.2.1 Linear attack

The proposed scheme is free from linear attack. Consider the equation below which is the tropical form of the classical linear attack as in (2).

$$\begin{aligned} X_1 \otimes X &= X \otimes X_1 \\ X_2 \otimes Y &= Y \otimes X_2 \\ X \otimes Y &= U \end{aligned} \tag{5}$$

equation (5) does not translate into a system of linear equations if U, X_1, and X_2 are known and X, Y are unknown. Moreover, there is no efficient algorithm to solve a system of linear equations in this form [8, 5]. Consider the public matrix $U = B$ shared by Salima as in the proposed scheme, that is

$$\mathrm{B}=\begin{bmatrix} 171 & 172 & 175 \\ 166 & 174 & 172 \\ 169 & 177 & 167 \end{bmatrix}$$

To solve (5), an attacker needs to find the following;

$$X = B \otimes Y^{\otimes -1}.$$

The inverse of Y can not be obtained in tropical semiring. In other words, the equation does not translate into a system of linear equations. Thus, X and Y can not be found such that $XDY = K$ is the shared secret key, where D is the public key of Salim.

Proposition 2. *The proposed scheme is free from linear attack.*

Proof. Let X_1, X_2, U be in $M_k(\mathbb{Z}_{min})$.
Then, X_1, X_2 and U are not invertible due to the structure of $M_k(\mathbb{Z}_{min})$, the tropical matrices in tropical semiring. The only invertible matrices are diagonal matrices and generalised permutation matrices. Thus, the system (5) is not solvable in $M_k(\mathbb{Z}_{min})$, since

$$X = U \otimes Y^{\otimes -1}$$

is not obtainable in $M_k(\mathbb{Z}_{min})$ if Y is not invertible.
Thus, the proposed scheme is free from linear attack.

□

4.2.2 Kotov and Ushakov attack

In Shpilrain's tropical key exchange, the private keys are polynomials $p_1(x), p_2(x)$ and the public keys are matrices a, B, U and V. Generally tropical polynomial is given by $P(x) = \oplus_{i=1}^{d} a_i \otimes x^{\otimes i}$, from the polynomial $p(a) = \oplus_{i=1}^{d} a_i \otimes a^{\otimes i}$ which is used in the cryptographic process of the scheme to compute K_a. In Kotov and Oshakov attack, they try to write (5) by writing $X = \oplus_{i=1}^{D} a_i \otimes a^{\otimes i}$ and $Y = \oplus_{j=1}^{d} b_j \otimes B^{\otimes j}$ in polynomial form, where i and j are chosen and defined from the degree d of the polynomial $p(x)$. In the attack, D was assumed to be a bigger number than the maximal degree of any tropical polynomial that could be used by the sender and receiver. It is clear from the proposed scheme that the tropical polynomial was not used in the scheme. Thus, the Kotov and Oshakov attack could not be easily applied to the proposed scheme.

4.2.3 Brute force attack

In terms of brute force attack, the proposed scheme has $Y_1, Y_2, Q \in CM_k(\mathbb{Z}_{min})$ which are randomly chosen private keys, unlike the original scheme that has only two natural numbers m and n. Thus, it takes time for an attacker to find all possible combinations of private keys. assuming $k \geq 10$ and elements of the matrices are randomly chosen between $[1, 1000]$. Thus, it takes time to find the possible combinations of the private keys as described by [8].
If a $k \times k$ matrix with $k = 10$ is used in the proposed scheme and its elements are chosen between $[1, 1000]$. The total number of different matrices can be formed in $1000k^2$. The complexity is $\mathcal{O}(1000k^2)$. For $k = 10$, the complexity is 1000^{100}, which is an astronomically large number. Thus, guessing the secret key K by brute force is computationally infeasible especially if k is large.

4.2.4 Comparison Between the Existing and Proposed Schemes

Table 4: Comparison between Existing and Proposed Schemes

S.No.	Stickel's Key exchange	Tropical Key exchange Scheme	Proposed Key exchange Scheme
1.	private keys are natural numbers	Private keys are polynomials	Private keys are matrices
2.	Computations on classical algebra	Tropical semiring	Tropical semiring
3.	Difficulty relies on DLP	Tropical semiring problem (TSP)	TSP and matrix power function problem
4.		Polynomial version was used to improve the scheme	Exponential version
5.	Tropical matrices	Periodicity of tropical matrices powers	No matrix powers
6.	Linear attack	Linear attack due to the polynomial	Free from linear attack

The table above shows the comparison between the three schemes, and the advantages of the proposed scheme against the original Stickel's scheme and tropical schemes.

5 Some Properties of The Structures Used in The Proposed Scheme

Below are some of the observed and derived propositions and corollaries as well as their proofs in tropical matrix algebra.

Proposition 3. *If* $a, B \in CM_k(\mathbb{Z}_{min})$ *then* $a \otimes B \in CM_k(\mathbb{Z}_{min})$.

Proof. Let $a = a_i = (a_0, a_1, \ldots, a_n)$ and $B = b_j = (b_0, b_1, \ldots, b_n)$, thus, $(\text{a} \otimes B)_{ij} = \oplus_{k=1}^{n}(a_{ik} \otimes b_{kj}) = \min(a_{ik} + b_{kj})$. Since a and B are circulant matrices, each row and column of a and B is a cyclic permutation of the row and column above it.

Let $(i, j)th$ elements of $(a \otimes B)_{ij} = \oplus_{(i,j)=1}^{n}(a_i \otimes B_j) = \min(a_i + B_j)$ where a_i is the ith row of a and B_j is the jth column of B. By the properties of a and B, each row a_i is a cyclic permutation of first row a_0 and each column B_j is a cyclic permutation of the first column B_0.Thus,

$$a_i = a_0 \text{ cyclically shifted by } i \text{ positions}$$
$$B_j = B_0 \text{ cyclically shifted by } j \text{ positions}$$

Therefore, $(i, j)th$ elements of $a \otimes B = \oplus_{(i,j)=1}^{n}(a_i \otimes B_j) = \min[(a_0 \text{ cyclically shifted by } i \text{ positions}) + (B_0 \text{ cyclically shifted by } j \text{ positions})]$. The sum will also be a cyclic permutation of $a_0 + B_0$. Thus, $(a \otimes B)_{ij} = \min(a_0 + B_0)$. This shows $a \otimes B$ has constant values in each row, thus, it is circulant.

□

Proposition 4. *Let $a \in M_k(\mathbb{Z}_{min})$ and $B \in CM_k(\mathbb{Z}_{min})$ then $a \otimes B \notin CM_k(\mathbb{Z}_{min})$.*

Proof. Let $a = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_7 & m_8 \end{bmatrix}, B = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_3 & m_1 & m_2 \\ m_2 & m_3 & m_1 \end{bmatrix}$ and $C = a \otimes B = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$

where $m_i, i = [1, n]$ are not equal. From the definition of tropical multiplication, we have $C_{ij} = (a \otimes B)_{ij} = \oplus_{k=1}^{n}(a_{ik} \otimes b_{kj}) = \min(a_{ik} + b_{kj})$. Now,

$$\begin{aligned} c_{11} &= \min(m_1 + m_1, m_2 + m_3, m_3 + m_2) \\ c_{22} &= \min(m_4 + m_2, m_5 + m_1, m_6 + m_3) \\ c_{33} &= \min(m_7 + m_3, m_8 + m_2, m_9 + m_1) \end{aligned}$$

to show that $a \otimes B \notin CM_k(\mathbb{Z}_{min})$, it is sufficient to show that $c_{11} \neq c_{22} \neq c_{33}$., since the diagonal entries of any circulant matrix are equal. From the above expressions of c_{11}, c_{22} and c_{33} it is clear that they are not equal, therefore, $a \otimes B \notin CM_k(\mathbb{Z}_{min})$.

□

Proposition 4 holds for $B \otimes a$ which is the special case of the proposition, thus, yields the corollary below.

Corollary 1. *Let $a \in M_k(\mathbb{Z}_{min})$ and $B \in CM_k(\mathbb{Z}_{min})$ then $B \otimes a \notin CM_k(\mathbb{Z}_{min})$.*

Proposition 5. *if $a, B \in M_k(\mathbb{Z}_{min})$ and $a^{\otimes B}$ is matrix power function, then $(a^{-1})^{\otimes B}$ will result to additive identity matrix (∞) of $M_k(\mathbb{Z}_{min})$, provided that the entry element of a and B are greater than 1 and a is invertible in $M_k(\mathbb{Z}_{min})$.*

Proof. Let $a = \begin{bmatrix} a_{11} & a_{12} \dots & a_{1n} \\ a_{21} & a_{22} \dots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} \dots & a_{nn} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \dots & b_{1n} \\ b_{21} & b_{22} \dots & b_{2n} \\ \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} \dots & b_{nn} \end{bmatrix}$

then

$$a^{-1} = \begin{bmatrix} -a_{11} & \infty \dots & \infty \\ \infty & -a_{22} \dots & \infty \\ \vdots & \vdots & \vdots \\ \infty & \infty \dots & -a_{nn} \end{bmatrix} \text{ and}$$

$$a^{-1 \otimes B} = \begin{bmatrix} -a_{11} & \infty \dots & \infty \\ \infty & -a_{22} \dots & \infty \\ \vdots & \vdots & \vdots \\ \infty & \infty \dots & -a_{nn} \end{bmatrix}^{\otimes} \begin{bmatrix} b_{11} & b_{12} \dots & b_{1n} \\ b_{21} & b_{22} \dots & b_{2n} \\ \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} \dots & b_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} -a_{11}^{b_{11}} \otimes \infty^{b_{21}} \otimes \dots \otimes \infty^{b_{n1}} & -a_{11}^{b_{12}} \otimes \infty^{b_{22}} \otimes \dots \otimes \infty^{b_{n2}} \dots & -a_{11}^{b_{1n}} \otimes \infty^{b_{2n}} \otimes \dots \otimes \infty^{b_{nn}} \\ \infty^{b_{11}} \otimes -a_{22}^{b_{21}} \otimes \dots \otimes \infty^{b_{n1}} & \infty^{b_{12}} \otimes -a_{22}^{b_{22}} \otimes \dots \otimes \infty^{b_{n2}} \dots & -\infty^{b_{1n}} \otimes -a_{22}^{b_{2n}} \otimes \dots \otimes \infty^{b_{nn}} \\ \vdots & \vdots & \vdots \\ \infty^{b_{11}} \otimes \infty^{b_{21}} \otimes \dots \otimes -a_{nn}^{b_{n1}} \dots & \infty^{b_{12}} \otimes \infty^{b_{22}} \otimes \dots \otimes -a_{nn}^{b_{n2}} \dots & \infty^{b_{1n}} \otimes \infty^{b_{2n}} \otimes \dots \otimes -a_{nn}^{b_{nn}} \end{bmatrix}$$

$$= \begin{bmatrix} -a_{11}b_{11} + \infty + \dots + \infty & -a_{11}b_{12} + \infty + \dots + \infty \dots & -a_{11}b_{1n} + \infty + \dots + \infty \\ \infty - a_{22}b_{21} + \dots + \infty & \infty - a_{22}b_{22} + \dots + \infty & \infty - a_{22}^{b_{2n}} + \dots + \infty \\ \vdots & \vdots & \vdots \\ \infty + \infty + \dots - a_{nn}b_{n1} & \infty + \infty + \dots - a_{nn}b_{nn} \dots & \infty + \infty + \dots - a_{nn}b_{nn} \end{bmatrix}$$

Since,

$$\infty^m = \infty \text{ for all } m > 1$$
$$\infty \pm m = \infty$$
$$\infty \times \pm m = \pm\infty \text{ if } m \neq 0$$
$$m^\infty = \infty \text{ if } m > 1$$

Thus, by the definition of $\otimes$ on tropical semiring we have $(a^{-1})^{\otimes B} = \begin{bmatrix} \infty & \infty \dots & \infty \\ \infty & \infty \dots & \infty \\ \vdots & \vdots & \vdots \\ \infty & \infty \dots & \infty \end{bmatrix}$ as claimed since the entry elements of a and B are greater than 1.

□

The corollary below is a special case of proposition 5.

Corollary 2. *if $a, B \in M_k(\mathbb{Z}_{min})$ and $a^{\otimes B}$ is matrix power function, then $(a)^{\otimes B^{-1}}$ will result to additive identity matrix (∞) of $M_k(\mathbb{Z}_{min})$, provided that the entry element of a and B are greater than 1 and B is invertible in $M_k(\mathbb{Z}_{min})$.*

6 Conclusions

Stickel's scheme was built upon a non-commutative structure using a matrix group as a platform. Tropical algebra was used as a platform to protect the scheme from linear attack and to improve its efficiency as tropical platforms play an important role in providing efficiency and less vulnerability to linear attack but the pattern displayed by the higher power of matrices (i.e., the number of operations executed to obtain the public key) makes tropical platforms susceptible to certain attacks. The improved scheme that was constructed using the matrix power function on a tropical semiring reduces the leakage of displaying patterns because matrices are replaced by natural numbers. Some properties of tropical matrices and tropical power functions were discovered. However, interested readers are implored to suggest possible ways of improving the security of the proposed scheme, as it is the trend for cryptographic protocols. There is often room for improvement. We suggest the use of matrix power function with other special forms of matrices on tropical semiring.

acknowledgments

This is to acknowledge the support provided by Universiti Sains Malaysia. This research is funded by the Universiti Sains Malaysia Research University (RU) Grant, account number, 1001/PMaTHS/8011121.

References

[1] K Ahmed, S Pal, and R Mohan. A review of the tropical approach in cryptography. *Cryptologia*, 47(1):63–87, 2023. https://doi.org/10.1080/01611194.2021.1994486.

[2] Arup Bose and Koushik Saha. *Random circulant matrices*. CRC Press, 2018. https://doi.org/10.1201/9780429435508-1.

[3] Peter Butkovič. *Max-linear systems: theory and algorithms*. Springer Science & Business Media, 2010. https://doi.org/10.1007/978-1-84996-299-5.

[4] Mariana Durcheva. *Semirings as building blocks in cryptography*. Cambridge Scholars Publishing, 2019.

[5] Mariana Ivanova Durcheva. Tres: Tropical encryption scheme based on double key exchange. *European Journal of Information Technologies and Computer Science*, 2(4):11–17, 2022. https://doi.org/10.24018/compute.2022.2.4.70.

[6] Jonathan S Golan. *Semirings and their Applications*. Springer Science & Business Media, 2013. https://doi.org.10.1007/978-94-015-9333-5.

[7] Jonathan S Golan and Jonathan S Golan. Semirings. *Semirings and Affine Equations over Them: Theory and Applications*, pages 1–26, 2003. https://doi.org/10.1007/978-94-017-0383-3_1.

[8] Dima Grigoriev and Vladimir Shpilrain. Tropical cryptography. *Communications in Algebra*, 42(6):2624–2632, 2014. https://doi.org/10.1080/00927872.2013.766827.

[9] Dima Grigoriev and Vladimir Shpilrain. Tropical cryptography ii: extensions by homomorphisms. *Communications in Algebra*, 47(10):4224–4229, 2019. https://doi.org/10.1080/00927872.2019.1581213.

[10] Steve Isaac and Delaram Kahrobaei. A closer look at the tropical cryptography. *International Journal of Computer Mathematics: Computer Systems Theory*, 6(2):137–142, 2021. https://doi.org/10.1080/23799927.2020.1862303.

[11] Delaram Kahrobaei, Ramón Flores, Marialaura Noce, et al. Group-based cryptography in the quantum era. *Not. Am. Math. Soc*, 70:2–13, 2023. https://doi.org/10.1090/noti2684.

[12] Matvei Kotov and Alexander Ushakov. Analysis of a key exchange protocol based on tropical matrix algebra. *Journal of Mathematical Cryptology*, 12(3):137–141, 2018. https://doi.org/10.1515/jmc-2016-0064.

[13] Diane Maclagan and Bernd Sturmfels. *Introduction to tropical geometry*, volume 161. American Mathematical Society, 2021. https://doi.org/10.1007/978-3-0346-0048-4_1.

[14] Any Muanalifah and Sergei Sergeev. On the tropical discrete logarithm problem and security of a protocol based on tropical semidirect product. *Communications in Algebra*, 50(2):861–879, 2022. https://doi.org/10.1080/00927872.2021.1975125.

[15] Karl Nachtigall et al. Powers of matrices over an extremal algebra with applications to periodic graphs. *Mathematical Methods of Operations Research*, 46(1):87–102, 1997.

[16] Dylan Rudy and Chris Monico. Remarks on a tropical key exchange system. *Journal of Mathematical Cryptology*, 15(1):280–283, 2020. https://doi.org/10.1515/jmc-2019-0061.

[17] Eligijus Sakalauskas. Enhanced matrix power function for cryptographic primitive construction. *Symmetry*, 10(2):43, 2018. https://doi.org/10.3390/sym10020043.

[18] Sergei Sergeev. Max algebraic powers of irreducible matrices in the periodic regime: An application of cyclic classes. *Linear Algebra and Its Applications*, 431(8):1325–1339, 2009. https://doi.org/10.1016/j.laa.2009.04.027.

[19] Vladimir Shpilrain. Cryptanalysis of stickel's key exchange scheme. In *International computer science symposium in Russia*, pages 283–288. Springer, 2008. https://doi.org/10.1007/978-3-540-79709-8_29.

[20] Eberhard Stickel. A new method for exchanging secret keys. In *Third International Conference on Information Technology and Applications (ICITA'05)*, volume 2, pages 426–430. IEEE, 2005. https://doi.org/10.1109/icita.2005.33.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 163–176.

An Improved Privacy-preserving Decision Tree Classifier based on Secret Sharing

Sheng-Hsin Tso[1] and Lo-Yao Ye[1]

National Central University, Taoyuan, Taiwan tsokiyo1@gmail.com,yehloyao@ncu.edu.tw

Abstract. This paper addresses the challenge of securely outsourcing a trained model to a server while preserving both the model owner's and the data owner's privacy. Previous works usually use public key cryptosystems such as homomorphic encryption or zero-knowledge proofs, which are usually considered to be inefficient with heavy computations. Inspired from Chang et. al.'s work which only use lightweight cryptosystem (ie., secret sharing), we introduce two new algorithms based on their original schemes. We will show how to further improve the efficiency of their schemes without sacrifice the security of the schemes.

Keywords: Beaver triple, secure comparison, decision tree classification, outsourcing, privacy preserving, secret sharing

1 Introduction

In today's rapidly advancing technological landscape, the proliferation of data and the need for efficient data analysis tools have given rise to significant advancements in machine learning and artificial intelligence. Decision tree learning [13, 5] is a type of analytical model used for making decisions and predicting outcomes. They are particularly useful because they allow users to visually explore data, making it easier to understand the relationships between different variables. The breakdown of the key concepts related to decision trees are as follows:

- Hierarchical Classification Structures: Decision trees are hierarchical, meaning they start with a broad decision at the "root" and branch out into more specific decisions, ultimately leading to a final outcome or classification at the "leaves."
- Independent and Dependent Variables: In decision trees, independent variables (also called attributes) are the factors that influence the outcome. The dependent variable (or class) is what you're trying to predict or classify.
- Nodes and Branches:
  - Root: The topmost node in a decision tree, where the first decision is made.
  - Nodes: These can be either:
    - Decision Nodes: Points in the tree where a decision is made based on a test applied to one of the independent variables. This divides the data into smaller subsets.
    - Terminal Nodes (Leaves): The end points of the tree, representing the final classification or decision.
  - Branches: Paths that connect the nodes, representing the sequence of decisions made based on the attributes.

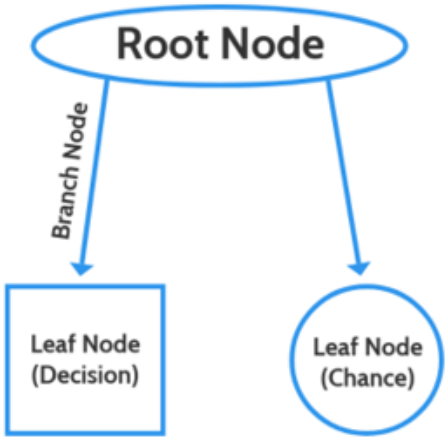


Figure 1: Example of a simple decision tree [13]

Algorithms based on tree structures are among the most popular methods in supervised learning due to their high accuracy and stability in generating predictive models. For instance, Classification and Regression Trees (CART) [10, 8] is a notable technique in this domain. This approach provides a clear and understandable way to visualize decision-making processes, making it a valuable tool for predictive analytics.

However, with the increasing reliance on cloud computing for storing and processing large-scale data, ensuring the privacy and security of sensitive information has become a paramount concern. For example, a significant challenge is outsourcing pre-trained models to servers while preserving model privacy [9]. Moreover, protecting data privacy from the server's perspective is equally crucial. Most current solutions rely on homomorphic encryption [14, 2] or zero knowledge proofs [1], which is often inefficient and unsuitable for IoT devices with limited computational capabilities. Recently, Chang et al. [3, 4] proposed a privacy-preserving method for decision tree classification. Their primary strategy involves using secret-sharing [12] between the model owner, the service user (data owner), and the cloud server. This approach facilitates secure comparison computations without revealing the model or the data. The cryptographic method they utilized involves lightweight secret-sharing, offering a more efficient alternative to more resource-intensive methods like homomorphic encryption. This makes it better suited for IoT environments that have lower storage and communication capabilities.

1.1 Motivation and Our Contribution

The motivation behind this research is driven by the growing need for secure and efficient methods to outsource machine learning models in the era of cloud computing and IoT. As organizations increasingly leverage cloud services to store and analyze data, the risk of exposing sensitive information to unauthorized parties escalates. Ensuring the confidentiality of both the model and the data during the outsourcing process is critical to maintaining trust and compliance with privacy regulations. Current solutions that rely on homomorphic encryption or zero-knowledge proofs, while secure, are not practical for widespread use due to their high computational demands. This inefficiency poses a significant barrier to the adoption of secure cloud-based machine learning services, particularly in environments with limited computational power.

To address these challenges, this paper introduces improved privacy-preserving decision tree classifiers based on secret sharing. Inspired by the work of Chang et al., which utilizes

lightweight cryptographic techniques, we propose two novel algorithms that enhance the efficiency of their schemes without compromising security. Our approach is particularly well-suited for IoT environments, where computational resources and communication capabilities are limited.

The technical details are explained as follows: Considering a model M structured as a tree, the objectives of the model owner are twofold: (1) to issue a token to a user, allowing them access to enjoy model M, and (2) to provide some limited information about model M. To maintain the privacy of M, we employ secret sharing techniques. This involves splitting each node in the tree into two shares. One share is given to the user, while the other is retained by the server. On the other hand, data owner also splits his/her data into shares and gives half of the shares to the server. This method ensures that a decision (of the decision three) can be made based on the shares of the model and the data without leaking the information of model (ie., attribute or feature of the model) as well as the information of the data of the data owner. In other words, based on the secure computation between the data owner and the server , a decision can be made while the privacy of the model as well as the data are securely preserved at the end of our protocols. Our second algorithm further guarantees that the secrecy of the data is still preserved even if the model owner and the server collude, leading to the server knowing the information (i.e., features or attributes) of a model.

2 Building Blocks

2.1 Pseudorandom Generator (PRG)

PRG is a fundamental concept in cryptography and computer science, used to generate sequences of numbers that appear random but are actually computed from a deterministic process. PRGs are crucial for various applications including cryptography, simulations, and randomized algorithms.

2.2 2-out-of-2 Secret Sharing

Secret sharing is a procedure for distributing a secret between a group of participants, each of whom is allocated a share of the secret. The secret can be recovered only when a sufficient number of shares of the secret. In this paper, only 2-out-of-2 secret sharing is used in our proposal. The security guarantees that only one out of two shares does not reveal the secret.

- Preparation Phase (offline): $SS(x) \rightarrow [x]_1, [x]_2$: A Dealer plits a secret x into two shares $[x]_1, [x]_2$ where $x, [x]_1, [x]_2 \in Z_q^*$. Here q is a large prime. Then, $[x]_i$ is sent to a participant U_i, $1 \leq i \leq 2$.
- Computation Phase (offline): $SS^{'}(x, [x]_b) \rightarrow [x]_{\bar{b}}$ on input of $x, [x]_b \in Z_q^*$. It is an alternative protocol of $SS(x)$ which allows a participant to split his/her secret x into $[x]_1, [x]_2 \in Z_q^*$. He/She holds, $[x]_b$ and sends $[x]_{\bar{b}}$ to the other participant. In other words, this phase can be done without a dealer's assistant.
- Recovery Phase (online): $Recover([x]_1, [x]_2) \rightarrow x$: Obtain x from $[x]_1, [x]_2$. With the joint work by the two participants, each presents his/her share $[x]_1, [x]_2$, respectively, then the original secret $x \in Z_q^*$ can be recovered.

2.3 Beaver Triples

Beaver triples [11] are a cryptographic tool used in secure multiparty computation (MPC), a type of cryptographic protocol that allows multiple parties to jointly compute a function

over their inputs while keeping those inputs private. The main purpose of Beaver's Triples is to facilitate secure and efficient multiplication of shared secrets, which is a challenging and critical operation in secure multiparty computation. In many MPC schemes, additions and multiplications are the primary operations needed to construct more complex computations. While addition of secret shares is relatively straightforward, multiplication introduces complexity because it can potentially leak information about the inputs.

- Preparation Phase (offline): Before the actual computation begins, a trusted third party or a distributed protocol generates triple of number (a, b, c) where a and b are random and $c = a \times b$. These values are then secretly distributed among the parties in such a way that no single party knows all of a, b, and c, but rather only holds a "share" of each.

- Computation Phase (online): When the parties need to compute the product of two shared secret, say x and y, they use the pre-distributed triples. Each party calculates the differences between their shares of x and a, and y and b, respectively, and these differences are made public. Since a and b are random and unknown to all parties, revealing these differences does not compromise the security of x and y.

- Using the Triples: Once the differences are known, each party can compute a share of xy using the equation:

$$xy = c + (x - a)b + (y - b)a + (x - a)(y - b)$$

Here, c is the precomputed product of a and b, and the terms involving $x - a$ and $y - b$ are known from the previous step. Each party then adjusts their share of c using the additional terms to get a share of the product xy.

The use of Beaver's triples significantly reduces the complexity and improves the security of performing multiplications in a distributed manner, making secure multiparty computation more feasible for practical applications.

3 Revisit of Chang et. al.'s Privacy-preserving Scheme

Chang et. al. introduced a privacy-preserving delegation of decision tree classification protocol in 2020 [3, 4]. The scheme allows the model owner to outsource the classification service to cloud server without leaking the content of the model. The cloud server is responsible for storing the model and providing classification services for potential users. The confidentiality of both input and output of the service is also maintained during the classification protocol. In addition, the data belonging to the model owner is also considered as sensitive information and will not be disclosed during the executing of the protocol. Their scheme utilizes secret sharing as well as the Beaver triples instead of homomorphic cryptosystems in order to improve the efficiency.

Their scheme consists of three entities: model owner (MO), Service User (SU) and Cloud Server (CS).

- The Model Owner (MO), who has the decision tree classification model $M.MO$ performs classification work using M where M must be preserved without leaking the information of M. Here M is protected by the 2-out-of-2 secret sharing.

- Service User (SU): SU is the machine learning service user possessing the source data $D.SU$ desires to predict the class of D using the model M without leaking any information on D.

- Cloud Server (CS): CS is assumed to have sufficient storage space and computation power. The purpose of CS is to store the share of model $[M]$ and respond to queries from SU during the process of classification service. In short, CS and SU in this protocol is in cooperate with each other to find the final classification result without knowing the model $[M]$ and CS doesn't know the data D.

Algorithm 1 Comparison Algorithm I

Input A: $x, [x]_1, [y]_1$ /* A is the Service User (SU)
Input B: $[x]_2, [y]_2$ /* B is the Cloud Server (CS)
Output B: 1 if $(x > y)$ or 0 if $(x \leq y)$
Process of A:
Randomly picks $\alpha \in Z_q^*$ where $x + a < q$, and $y + a < q$.
Executes $SS(\alpha) \rightarrow [\alpha]_1, [\alpha]_2$.
Sends$[\alpha]_2$ to B.
Computes $[s]_1 = [x]_1 + [\alpha]_1$, and $[h]_1 = [y]_1 + [\alpha]_1$.
Process of B:
Computes $[s]_2 = [x]_2 + [\alpha]_2$ and $[h]_2 = [y]_2 + [\alpha]_2$.
Process of A:
Send $[s]_1$ and $[h]_1$ to B.
Process of B:
Executes $Recover([s]_1, [s]_2) \rightarrow s$.
Executes $Recover([h]_1, [h]_2) \rightarrow h$.
Outputs 1 if $s > h$, and 0 otherwise.
Sends comparison result (ie.,1 or 0) to A.

Algorithm 1 is a secure comparison protocol in a two-party setting. In this algorithm, the model information y (in the application of a decision tree, y can be a vector $y = (y_1, y_2, \ldots, y_n)$ where each y_i represent a value of a corresponding node) is divided by the model owner MO at the beginning and the corresponding shares of y is $[y]_1, [y]_2$. At the initial phase, A possesses the plaintext of x, $[x]_1, [y]_1$ and shares the value $[x]_2$ with B. B also has the share $[y]_2$ received from MO. With this algorithm, A and B can compare two shared values without knowledge of y, and B does not know the value of the data x. At the end, when B sends the comparison result to A, then both entities can proceed to the next step for the comparison of the next layer of the decision tree.

Instead of Algorithm 1, Chen et. al. also proposed a relatively complexed version [3] which combines a Beaver triple with the secret sharing scheme to do the same comparison. The detailed algorithm is described in Algorithm 2 and Algorithm 3.

In Algorithm 1, $s = x + \alpha$ and $h = y + \alpha$. So $s - h = x - y$. $x > y$ if $s - h > 0$. Otherwise, $x \leq y$. On the other hand, in Algorithm 2, $s = zr_1 + r_2 = (x - y + \ell)r_1 + r_2$, and $h = \ell r_1 + r_2$. So $s - h = (x - y)r_1$. The same as Algorithm 1, if $s - h > 0$, then $x > y$. Otherwise, $x \leq y$. Algorithm 2 further guarantees that the disclosure of x will not harm the secrecy of y, and vice versa because of the random value r_1 is unknown to B.

4 Proposed Schemes

Before formally introducing out new schemes, we first loosen the requirement of the underlying secret sharing scheme. That is, we here choose random values from integer domain Z instead of chosen from Z_q^*. This implies that each share may be a positive integer or an negative integer. Details is described below:

- Preparation Phase (offline): $SS(x) \rightarrow [x]_1, [x]_2$: A Dealer splits a secret x into

Algorithm 2 Comparison Algorithm II

Input A: $x, [x]_1, [y]_1$ where $x, y \leq \ell$ /* A is the Service User (SU)
Input B: $[x]_2, [y]_2$ where $x, y \leq \ell$ /* B is the Cloud Server (CS)
Output B: 1 if $(x > y)$ or 0 if $(x \leq y)$
A or B produces $[\ell]_1, [\ell]_2$.
A generates $[z]_1 = [x]_1 - [y]_1 + [\ell]_1$.
B generates $[z]_2 = [x]_2 - [y]_2 + [\ell]_2$.
A picks two positive integers r_1, r_2 where $2\ell r_1 + r_2 < q$.
A generates $[r_1]_1, [r_1]_2, [r_2]_1, [r_{2]}]_2$.
By utilizing the Beaver triple algorithm (Algorithm 3), A and B perform the following steps:
A computes $[s]_1 = [zr_1 + r_2]_1$, and $[h]_1 = [\ell r_1 + r_2]_1$.
B computes $[s]_2 = [zr_1 + r_2]_2$, and $[h]_2 = [\ell r_1 + r_2]_2$.
A Send $[s]_1$ and $[h]_1$ to B.
B reconstructs s and h.
B outputs 1 if $s > h$, and 0 otherwise.
B sends comparison result (ie., 1 or 0) to A.

Algorithm 3 Multiplication Algorithm (Beaver Triple)

Input A: $x, [x]_1, [y]_1$
Input B: $[x]_2, [y]_2$
Output A: $[xy]_1$
Output B: $[xy]_2$
A gets three shares $[a]_1, [b]_1, [ab]_1$ from MO.
B gets three shares $[a]_2, [b]_2, [ab]_2$ from MO.
A sends $[x]_1 - [a]_1$ to B.
B sends $[x]_2 - [a]_2$ to A.
A or B produces $\varepsilon = x - a$, individually.
A sends $[y]_1 - [b]_1$ to B.
B sends $[y]_2 - [b]_2$ to A.
A or B produces $\rho = y - b$, individually.
A computes $[xy]_1$ as $[xy]_1 = [ab]_1 + \varepsilon[b]_1 + \rho[a]_1 + \varepsilon\rho$.
B computes $[xy]_2$ as $[xy]_2 = [ab]_2 + \varepsilon[b]_2 + \rho[a]_2 + \varepsilon\rho$.

two shares $[x]_1, [x]_2$ where x is a positive integer, and $[x]_1, [x]_2 \in Z$, such that $x = [x]_1 + [x]_2$. Then, $[x]_i$ is sent to a participant U_i, $1 \leq i \leq 2$.

- Computation Phase (offline): $SS^{'}(x, [x]_{\bar{b}}) \rightarrow [x]_{\bar{b}}$ Output $[x]_{\bar{b}} \in Z$, on input of $x, [x]_b$.It is an alternative protocol of $SS(x)$ which allows a participant to split his/her secret x into $[x]_1, [x]_2$. He/She holds, $[x]_b$ and sends $[x]_{\bar{b}}$ to the other participant. In other words, this phase can be done without a dealer's assistant.

- Recovery Phase (online):Recover $([x]_1, [x]_2) \rightarrow x$: Obtain x from $[x]_1, [x]_2$. With the joint work by the two participants, each presents his/her share $[x]_1, [x]_2$, respectively, then the original secret s can be recovered.

4.1 New Algorithm 1 for Secure Comparison

Algorithm 4 Comparison Algorithm I*

Input A: $x, [x]_1, [y]_1$ /* A is the Service User (SU)
Input B: $[y]_2$ /* B is the Cloud Server (CS)
Output B: 1 if $(x > y)$ or 0 if $(x \leq y)$
Process of A:
Randomly picks $\alpha \in Z$.
Computes $[s]_1 = [x]_1 + [y]_1 + \alpha \in Z$, and $[s]_2^{'} = [x]_2 - \alpha \in Z$.
Sends $[s]_1$ and $[s]_2^{'}$ to B.
Process of B:
Computes $[s]_2 = [s]_2^{'} - [y]_2$.
Executes $Recover([s]_1, [s]_2) \rightarrow s$.
Outputs 1 if $s > 0$, and outputs 0 otherwise.
Sends comparison result (ie., 1 or 0) to A.

4.2 New Algorithm 2 for Secure Comparison

Algorithm 5 Comparison Algorithm II*

Input A: $x, [x]_1, [y]_1$ /* A is the Service User (SU)
Input B: $[x]_2, [y]_2$ /* B is the Cloud Server (CS)
Output B: 1 if $(x > y)$ or 0 if $(x \leq y)$
A produces $[\ell]_1, [\ell]_2 \in Z$ where $\ell = [\ell]_1 + [\ell]_2 \in Z^+$. A then sends $[\ell]_2$ to B.
A generates $[x - y]_1 = [x]_1 - [y]_1$.
B generates $[x - y]_2 = [x]_2 - [y]_2$.
By utilizing the Beaver triple algorithm (Algorithm 3), A and B perform the following steps in an interactive way:
A computes $[s]_1 = [(x - y)\ell]_1$.
B computes $[s]_2 = [(x - y)\ell]_2$.
A sends $[s]_1$ to B.
B reconstructs $s = (x - y)\ell$.
B outputs 1 if $s > 0$, and 0 otherwise.
B sends comparison result (ie., 1 or 0) to A.

Comparing with the New Algorithm 1, our New Algorithm 2 is more secure since the data x remains secret even if the server knows y. This is because that the server B only knows $s = (x - y)\ell$ at the end without the knowledge of x and ℓ even if y is revealed.

4.3 Toy Examples

For easy to understand and to show the correctness, we give some toy examples for the proposed New Algorithm 1, and New Algorithm 2, respectively. Examples are given between the symbols $/^* \ldots ^*/$.

Example for Algorithm I*:

Algorithm 6 Example of Comparison Algorithm I*

1: **Input** A: $x, [x]_1, [y]_1$ $/^* x = 15, [x]_1 = 5, [x]_2 = 10, [y]_1 = 7 ^*/$
2: **Input** B: $[y]_2$ $/^* [y]_2 = -3 ^*/$
3: **Output** B: 1 if $(x > y)$ or 0 if $(x \leq y)$ $/^*$ output 1 cause $x = 15, y = 4$, and $x > y ^*/$
4: Process of A:
5: Randomly picks $\alpha \in Z$. $/^* \alpha = 8 ^*/$
6: Computes $[s]_1 = [x]_1 + [y]_1 + \alpha \in Z$, and $[s]_2^{'} = [x]_2 - \alpha \in Z$. $/^* [s]_1 = 5 - 7 + 8 = 6, [s]_2^{'} = 10 - 8 = 2 ^*/$
7: Sends $[s]_1$ and $[s]_2^{'}$ to B. $/^* [s]_1 = 6, [s]_2^{'} = 2 ^*/$
8: Process of B:
9: Computes $[s]_2 = [s]_2^{'} - [y]_2$. $/^* [s]_2 = 2 - (-3) = 5 ^*/$
10: Executes $Recover([s]_1, [s]_2) \rightarrow s$. $/^* ([s]_1 = 6, [s]_2 = 5) \rightarrow s = 11 ^*/$
11: Outputs 1 if $s > 0$, and outputs 0 otherwise.
12: Sends comparison result (ie., 1 or 0) to A.

We can see from the example that $s = 11 = 15 - 4 = x - y$. So the output of New Algorithm 1 is correct.

Example for Algorithm II*:

Algorithm 7 Comparison Algorithm II*

1: **Input** A: $x, [x]_1, [y]_1$ $/^* x = 15, [x]_1 = 5, [y]_1 = 7 ^*/$
2: **Input** B: $[x]_2, [y]_2$ $/^* [x]_2 = 10, [y]_2 = -3 ^*/$
3: **Output** B: 1 if $(x > y)$ or 0 if $(x \leq y)$ $/^*$ output 1 cause $x = 15, y = 4$, and $x > y ^*/$
4: A produces $[\ell]_1, [\ell]_2 \in Z$ where $\ell = [\ell]_1 + [\ell]_2 \in Z^+$. A then sends $[\ell]_2$ to B. $/^* \ell = [\ell]_1 + [\ell]_2 = 3 + 6 ^*/$
5: A generates $[x - y]_1 = [x]_1 - [y]_1$. $/^* [x - y]_1 = [x]_1 - [y]_1 = 5 - 7 = -2 ^*/$
6: B generates $[x - y]_2 = [x]_2 - [y]_2$. $/^* [x - y]_2 = [x]_2 - [y]_2 = 10 - (-3) = 13 ^*/$
7: By utilizing the Beaver triple algorithm (Algorithm 3), A and B perform the following steps in an interactive way:
8: A computes $[s]_1 = [(x - y)\ell]_1$. $/^* [s]_1 = 65$ via the Beaver triple $^*/$
9: B computes $[s]_2 = [(x - y)\ell]_2$. $/^* [s]_2 = 34$ via the Beaver triple $^*/$
10: A sends $[s]_1$ to B. $/^* [s]_1 = 65 ^*/$
11: B reconstructs $s = (x - y)\ell$. $/^* s = 11 * 9 = 99 ^*/$
12: B outputs 1 if $s > 0$, and 0 otherwise.
13: B sends comparison result (ie., 1 or 0) to A.

4.4 Actual Working Example and Its Implementation

With the knowledge of our new algorithms as well as the toy examples, we now give an actual working example of corrosion prediction based on a decision tree [7], and show how it can be done by our New Algorithm 1 to protect the sensitive data for the data owner and for the model owner, respectively.

In Fig. 2, the tree includes attributes such as: Depth Under the Sea Surface, Type of Stainless Steel Alloy, Exposure Time to Seawater, Maximum Depth of Pitting, and Depth

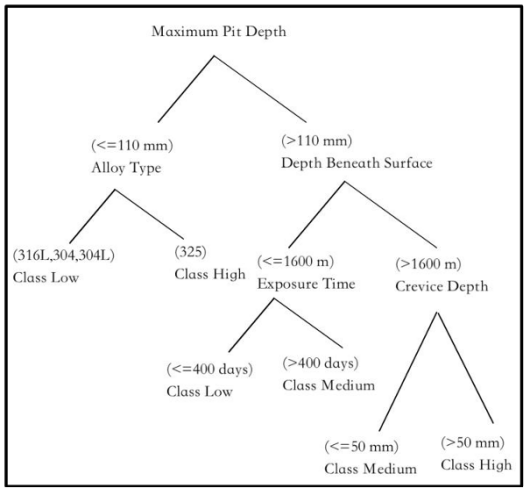


Figure 2: A decision tree for corrosion prediction [6]

of Crevice Formation. The class variable (what the data owner are predicting) could be the level of corrosion (ie., Low, Medium, or High). When classifying a new observation (for example, a stainless steel sample with specific characteristics), it start at the root of the tree (e.g., Maximum Pit Depth) and follow the branches based on the observation's attributes until it reach a terminal node. For example, if a sample has a Maximum Pit Depth of $120mm$, a Depth Beneath Surface of $1400m$, and an Exposure Time of 100 days, the decision tree will then classify it as having a Low level of corrosion.

We now implement this decision tree in Java language using our New Algorithm 1 for secure computation. In this implementation, the model owner has all the numbers of attributes or features of the tree (denotes as y) and a data owner (ie., the service user) with secret input x, desires to predict the level of corrosion of x (ie., the class of x) using the model without leaking x. Here h denotes the height of the tree, Fig. 3 is the source code of our New Algorithm 1, Fig. 4 is the detailed information of Input A (ie., x) and Input B (ie., y), respectively as well as the corresponding shares, and Fig. 5 is the output of the decision tree in each round (eg., 3 rounds if $h = 3$).

The label at that terminal node (e.g., Low corrosion) becomes the predicted outcome for this input $x = (120, 1000, 70)$.

5 Threat Model and Security Reductions

5.1 Threat Model

The definition of threat model follows Chang et. al.'s definition. That is, we here assume that all the entities are semi-honest. In other words, they are curious-but-honest adversaries that would strictly follow the protocol on one hand, but they are also interested in all data and attempt to learn more information during the execution of the protocol.

Based on the semi-honest adversary model, two security goals are defined: *Model Secrecy:It is required that the Service User (SU) and Cloud server (CS) learns nothing about the Model y.*
Data Secrecy: It is required that the Model Owner (MO) and Cloud Server (CS) learns nothing about the data x owned by the Service User (SU).

```
import java.util.Random;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of iterations (n): ");
        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            System.out.println("Iteration " + (i + 1) + ":");

            System.out.print("Enter the private data of data owner: ");
            int x = scanner.nextInt();
            System.out.print("Enter the private data of model owner:
");
            int y = scanner.nextInt();

            System.out.print("Enter part of the numbers from the data
owner: ");
            int x1 = scanner.nextInt();
            System.out.print("Enter the other part of the numbers from
the data owner: ");
            int x2 = scanner.nextInt();
            System.out.print("Enter part of the numbers from the model"
                    + " owner: ");
            int y1 = scanner.nextInt();
            System.out.print("Enter the other part of the numbers from
the model owner: ");
            int y2 = scanner.nextInt();

            Random rand = new Random();
            int alpha = rand.nextInt();

            int s1 = x1 - y1 + alpha;
            int s2Prime = x2 - alpha;
            int s2 = s2Prime - y2;
            int s = recover(s1, s2);

            System.out.println("alpha: " + alpha);
            System.out.println("[s]_1: " + s1);
            System.out.println("[s]'_2: " + s2Prime);
            System.out.println("[s]_2: " + s2);
            System.out.println("s: " + s);

            int result = (s > 0) ? 1 : 0;
            System.out.println("Result: " + result);
            System.out.println("-------------------------");
        }

        scanner.close();
    }

    public static int recover(int s1, int s2) {
        return s1 + s2;
    }
}
```

Figure 3: Implementation of New Algorithm 1

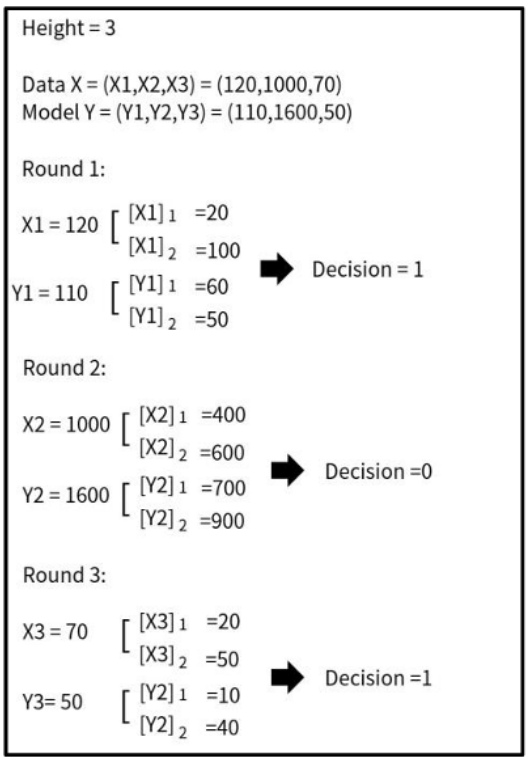


Figure 4: Information of input data x, model y and shares

5.2 Security Reductions

Chang et., al. have proved the security of their proposed schemes. We here use the reduction techniques to show that our new schemes inherits the security requirements of their schemes.

Firstly, for the model secrecy, since a model y is split into shares $[y]_1$ and $[y]_2$ using the 2-out-of-2 secret sharing before the execution of the protocol, the secrecy of y is preserved if CS and SU do not collide with each other. We hereafter will only consider the data secrecy against the Cloud Server (CS).

Theorem 1. *(Data Secrecy of the New Algorithm 1): The new algorithm 1 preserves the secrecy of the data against the internal adversary (ie., CS) if the data secrecy of Chang et. al.'s algorithm 1 is preserved.*

Proof. Let's look at the view of CS in each scheme. In New Algorithm 1, the view of CS is $\{[y]_2, [s]_1, [s]_2^{'}\}$, where $[s]_1 = [(x-y)\ell]_1$ and $[s]_2 = [x]_2 - \alpha$. On the other hand, in their original algorithm 1, the view of CS is $\{[x]_2, [y]_2, [s]_1, [h]_1, [s]_2, [h]_2\}$ where $[s]_1 = [x]_1 + [\alpha]_1$, $[h]_1 = [y]_1 + [\alpha]_1$, $[s]_2 = [x]_2 + [\alpha]_2$, and $[h]_2 = [y]_2 + [\alpha]_2$. We can always construct the CS view of our new algorithm based on the view of their algorithm. That is, do the following steps based on the view of their algorithm.

- Picks a random number α^*.
- Compute $[s]_1^* = [s]_1 = [h]_1 + \alpha^*$.
- Compute $[s]_2^{'*} = [x]_2 - \alpha^*$.

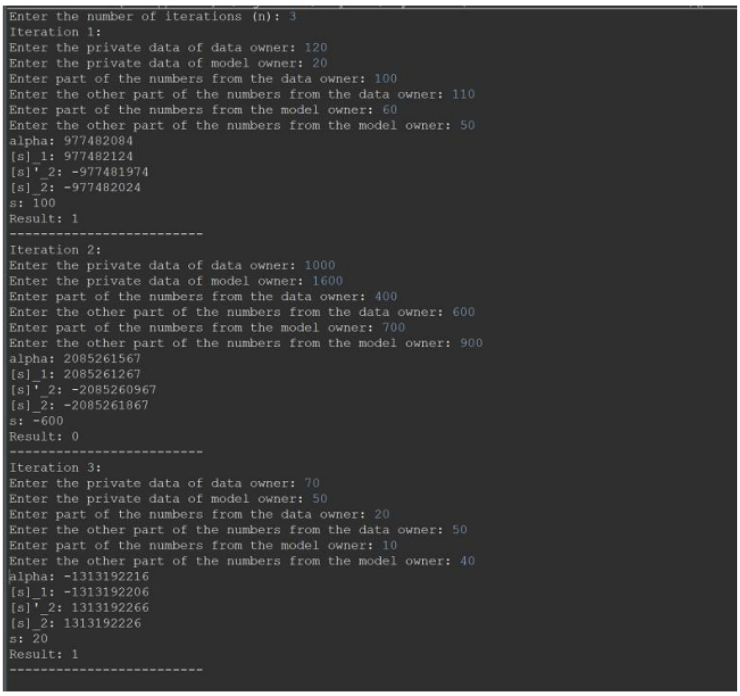


Figure 5: Decision result of input based on Figure 4

Since $[s]_1^* = [s]_1 - [h]_1 + \alpha^* = [x]_1 - [y]_1 + \alpha - [y]_1 - [\alpha]_1 + \alpha^* = [x]_1 - [y]_1 + \alpha^*$, and $[s]_2'^* = [x]_2 - \alpha^*$, we know that the CS of their algorithm can have the same view as the CS of our new algorithm. Therefore, if any CS (ie., the internal adversary) who can break the data secrecy of our new scheme, then we can use the adversary to break the data secrecy of Chang et., al.'s scheme.

□

Theorem 2. *(Data Secrecy of the New Algorithm 2): The new algorithm 2 preserves the secrecy of the data against the internal adversary (ie., CS) if the data secrecy of Chang et. al.'s algorithm 2 is preserved.*

Proof. Let's look at the view of CS in each scheme. In New Algorithm 2, the view of CS is $\{[x]_2, [y]_2, [s]_1, [s]_2, [\ell]_2\}$, where $[s]_1 = [(x-y)\ell]_1$ and $[s]_2 = [(x-y)\ell]_2$. On the other hand, in their original algorithm 2, the view of CS is $\{[x]_2, [y]_2, [z]_2, [s]_1, [h]_1, [s]_2, [h]_2\}$ where $[z]_2 = [x]_2 - [y]_2 + [l]_2, [s]_1 = [zr_1 + r_2]_1, [h]_1 = [\ell r_1 + r_2]_1, [s]_2 = [zr_1) + r_2]_2$, and $[h]_2 = [\ell r_1 + r_2]_2$ We can always construct the CS view of our new algorithm based on the view of their algorithm. That is, do the following steps based on the view of their algorithm.

- Compute $s^* = [s]_1 + [s]_2$.
- Compute $h^* = [h]_1 + [h]_2$.
- Compute $[s]^* - h^* = (x-y)r_1$.

Since $s^* - h^* = (x-y)r_1 = [(x-y)r_1]_1 + [(x-y)r_1]_2 = [s]_1^* + [s]_2^*$ by the 2-out-of-2 secret sharing, and $[(x-y)r_1]_2$ (ie., $[s]_2^*$) can be computed by the Beaver triple (ie., Algorithm 3) with B's input $\{[(x-y)]_2, [r_1]_2\}$. Here $[(x-y)]_2 = [x]_2 - [y]_2$. In addition, $[s]_1^* = s^* - h^* - [(x-y)r_1]_2 = [(x-y)r_1]_1$. So the view $\{[x]_2, [y]_2, [s]_1^*, [s]_2^*, [r]_2\}$ of CS of their

algorithm is exactly the same as the view of CS our new algorithm $\{[x]_2, [y]_2, [s]_1, [s]_2, [\ell]_2\}$. Therefore, if any CS (ie., the internal adversary) who can break the data secrecy of our new scheme II, then we can use the adversary to break the data secrecy of Chang et., al.'s scheme II.

□

5.3 Performance Comparison

Since our idea is inspired from Chang et. al. 's scheme, and the main purpose is to improve the efficiency of their scheme without sacrifice the security. So we here only compare the performance of our scheme with that of their scheme.

We here assume that each parameter consists of t bits. In their algorithm 1, and for each comparison, the Service User A possessing the sensitive data needs to send three parameters (ie., $[x]_2, [s]_1$ and $[h]_1$) to the Cloud Server B, so totally $3t$ bits is transmitted for one comparison. If the depth of the decision tree is d, then d times of comparison is required so the communication cost will become $3dt$ bits. On the other hand, in our New algorithm 1, and for each comparison, A needs to transmit only $[s]_1$ and $[s]_2^{'}$ to B which consists $2t$ bits, so d rounds will cost $2dt$ bits for communication.

It is exactly in the same case for the comparison of our New Algorithm 2 with their Algorithm 2. $[x]_2, [s]_1$ and $[h]_1$ are transmitted from A to B for one comparison in their scheme and only $[x]_2$ and $[s]_1$ are transmitted in our New Algorithm 2.

We omitted the comparison of computation cost since both schemes are based on simple secret sharing schemes without heavy computation. However, we can also see that our scheme is superior to their scheme even in the computation cost.

Table Head	**Data for transmission from A to B**		
	Data	***Size (one round)***	***Size (d round with depth d of the decision tree)***
Algorithm 1 [9]	$[x]_2, [s]_1, [h]_1$	*3t bits*	*3dt bits*
Algorithm 2 [10]	$[x]_2, [s]_1, [h]_1$	*3t bits*	*3dt bits*
Proposed 1	$[s]_1$, $[s]'_2$	*2t bits*	*2dt bits*
Proposed 2	$[x]_2$ $[s]_1$	*2t bits*	*2dt bits*

Figure 6: Performance Comparison

6 Conclusion

In this research article, we introduce two novel privacy-preserving decision tree evaluation systems based on Chang et. al.'s schemes using secret sharing techniques. The advantages of their scheme is the lightweight computation since no heavy computation is required comparing with those using homomorphic encryptions. We improved their scheme by efficiency without sacrifice the security. In these schemes, model owners can deploy their decision tree models onto cloud servers, enabling them to offer classification services to users of these cloud services. Our system ensures that no unauthorized parties gain access to the model's details. Additionally, it protects the confidentiality of users' query data and the results of their evaluations. Our proposed scheme enhances security for both the model and user data, while also significantly reducing the communication burden on data

owners, making it highly suitable for 5G and IoT environments. Our second algorithm further guarantees that the secrecy of the data is still preserved even if the model owner and the server collude, leading to the server knowing the information (i.e., features or attributes) of a model. Finally, the constructions in the paper is not a generic construction so it can only work for decision tree structures. Modification is required if it is used for other models. It can be considered as part of our future work.

References

[1] Haider Ali, Shafiqul Abidin, and Mahfooz Alam. Auditing of outsourced data in cloud computing: An overview. In *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 111–117. IEEE, 2024.

[2] Manuel Barbosa, Dario Catalano, and Dario Fiore. Labeled homomorphic encryption: Scalable and privacy-preserving processing of outsourced data. In *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I 22*, pages 146–166. Springer, 2017.

[3] Che-Chia Chang, Jian-Feng Lin, Song-Yi Hsu, and Yu-Chi Chen. Privacy-preserving delegation of decision tree classification. In *2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*, pages 1–2. IEEE, 2020.

[4] Che-Chia Chang, Jian-Feng Lin, Song-Yi Hsu, and Yu-Chi Chen. Privacy-preserving outsourcing decision tree evaluation from homomorphic encryption, 2020.

[5] Vinícius G Costa and Carlos E Pedreira. Recent advances in decision trees: An updated survey. *Artificial Intelligence Review*, 56(5):4765–4800, 2023.

[6] Rachel Cravit. What is a decision tree? how to make one with examples, 2024.

[7] CSense. About decision trees, 2024.

[8] Gururaj S Kori and Mahabaleshwar S Kakkasageri. Classification and regression tree (cart) based resource allocation scheme for wireless sensor networks. *Computer Communications*, 197:242–254, 2023.

[9] Ping Li, Jin Li, Zhengan Huang, Chong-Zhi Gao, Wen-Bin Chen, and Kai Chen. Privacy-preserving outsourced classification in cloud computing. *Cluster Computing*, 21:277–286, 2018.

[10] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[11] Pille Pullonen et al. Actively secure two-party computation: Efficient beaver triple generation. *Instructor*, 2013.

[12] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[13] Shan Suthaharan and Shan Suthaharan. Decision tree learning. *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*, pages 237–269, 2016.

[14] Kexin Xu, Benjamin Hong Meng Tan, Li-Ping Wang, Khin Mi Mi Aung, and Huaxiong Wang. Privacy-preserving outsourcing decision tree evaluation from homomorphic encryption. *Journal of Information Security and Applications*, 77:103582, 2023.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 177–187.

Lightweight and Privacy-Preserving Public-Key Authenticated Encryption with Keyword Search using Type-3 Pairing

Evon Yee-Ting Ng, Swee-Huay Heng, Syh-Yuan Tan and Koon-Ming Chan

Faculty of Information Science and Technology, Multimedia University, Melaka, Malaysia
evonngyt1602@gmail.com,shheng@mmu.edu.my,sytansy@gmail.com,koonming1996@gmail.com

Abstract. Cloud services and applications available anywhere in this modern era have increased the demand of users to store data on the cloud, thereby requiring encrypted storage to secure sensitive data. With encryption applied to the stored data, advanced mechanisms for searching over encrypted data without decryption are necessary to preserve data privacy, especially in scenarios where third-party cloud service providers are involved. If the keywords used for searching data and the search results are known by the cloud server, the user's privacy may be leaked. Hence, a searchable encryption scheme such as Public-key Authenticated Encryption with Keyword Search (PAEKS) needs to be implemented for confidential searching without decryption. In this paper, we upgrade the performance of a lightweight PAEKS scheme that is secure against keyword guessing attacks by reconstructing the scheme in Type-3 pairing. Subsequently, we benchmark the performance gained for the lightweight PAEKS scheme using SS1024 and BN254 curves for Type-1 and Type-3 pairings, respectively.

Keywords: Public-key Authenticated Encryption with Keyword Search · lightweight · Type-3 pairing · keyword guessing attack · multi-ciphertext indistinguishability

1 Introduction

In the current era, human daily life is bound tightly with the digital world, using technologies to fulfil different needs, one of which is cloud computing. Storing data on the cloud can save up local storage and provide ease of sharing with other users around the world, however, cloud data is exposed to the risk of being compromised due to data breaches, data interception, unauthorised access and other cybersecurity issues. Hence, data has been transformed to an unreadable format through encryption before transferring over the Internet to a cloud server and stored on cloud storage, thereby providing data confidentiality. However, with the encryption applied to the data, searching for data is more troublesome. The traditional searching method requires the decryption of data before performing a search, this results in a data privacy breach to the third party who performs searching. In the application of email service, a sender sends an email to a receiver through an email server and stores the email on cloud storage. The receiver receives emails from multiple senders, he or she may need to search an email content through a keyword of the email. Not only for email applications, cloud storage such as Google Drive also provides search functionality for data consumers to search for a shared file from data senders. To provide confidentiality and efficiency, various techniques of data searching in encrypted form have been invented such as Fully Homomorphic Encryption, Symmetric Searchable Encryption and Asymmetric Searchable Encryption or Public-key Encryption with Keyword Search (PEKS).

PEKS [4] is an asymmetric searchable encryption scheme that utilises public key cryptography to perform search functionality. In PEKS, the cloud server will search encrypted data without knowing anything about the keywords used to search and the content of search results. Like public key cryptography, it involves the public key and private key pair of receivers. Typically, three entities are involved in the system: sender, receiver, and cloud server. The sender is the owner or creator of the message, and he or she will select the keywords for the message. The receiver will request a search by providing a trapdoor containing the keywords to be searched. After searching, the receiver will decrypt the results to obtain the original message. In addition to storing the ciphertext of the message and keywords, the cloud server also performs searches and returns the corresponding message to the recipient. PEKS achieves ciphertext indistinguishability (CI) which is secure against chosen-plaintext attack (CPA), where an attacker cannot distinguish between two ciphertexts to reveal the keywords without knowing the corresponding trapdoor. However, it suffers from keyword guessing attack (KGA). In addition, PEKS does not possess trapdoor indistinguishability (TI), so an attacker can distinguish between two trapdoors to determine whether they contain similar keywords.

There are two types of KGA which are inside KGA (IKGA) and outside KGA (OKGA) [5], which is further divided into offline and online KGA. IKGA is performed by an inside attacker who has access to the receiver's trapdoor such as a cloud server. OKGA is performed by an outsider who is an eavesdropper over the communication channel to steal the trapdoor that is transmitted between the receiver and server. Due to the keywords for searching being frequently used words and short in length, brute force attacks can happen easily. An attacker has access to run all algorithms, except Trapdoor, as it needs the private key of the receiver. The attacker can run the PEKS algorithm repeatedly to generate ciphertexts for any keyword. Using the Test algorithm, the attacker can try to input the generated ciphertexts and trapdoor from the receiver to guess which keyword the receiver is searching for. If the Test algorithm outputs 1, the attacker will confirm the keyword in the input ciphertext matching with the keyword contained in the receiver's trapdoor. Therefore, the success of KGA by the adversary is due to two reasons: (1) the receiver's trapdoor is accessible and (2) the Test algorithm is freely runnable. By knowing the keywords that users are searching for, attackers can roughly guess the relevant content in the message, leading to data breaches and privacy leaks.

To prevent IKGA, PAEKS [9] is one of the solutions. Its main difference from the standard PEKS is it not only generates a key pair for receiver, but also for the sender. This provides authentication property by requiring the private key of the sender when performing the keyword encryption algorithm and requiring the sender's public key when performing the Trapdoor algorithm. This is to ensure only the private key holder which is the sender will be allowed to run the keyword encryption algorithm, thereby preventing any malicious party from freely generating the ciphertext of keywords to perform IKGA. Even if they have access to the algorithm, the ciphertext generated is not authenticated.

Most of the PEKS and PAEKS achieve CI only in a single-challenge ciphertext setting, where the adversary only needs to differentiate between ciphertexts of two single keywords without knowing the trapdoors of both keywords. If taking into account real-life situations where each message or file contains multiple keywords, two messages or files may have some similar keywords, but the data owners do not want this information to be known by any third party. To capture a chosen multi-keyword attack in which the adversary can distinguish whether two encrypted data contain the same keywords, MCI [16] is needed. Keyword searchable encryption scheme that achieves CI is still vulnerable to KGA because an efficient algorithm is found to be able to distinguish two ciphertexts containing the same keyword. With MCI, attackers cannot distinguish between ciphertexts of two sets of keywords, therefore improving the security of the scheme.

Bilinear pairing is commonly used in PEKS but it requires heavy computational

Table 1: Comparison of Pu et al.'s Lightweight PAEKS Scheme with Pioneering Schemes.

Scheme	PAEKS [9]	PAEKS-MCI [16]	Lightweight PAEKS [15]
Authentication	✓	✓	✓
IKGA	✓	✓	✓
OKGA	✗	✗	✗
MCI	✗	✓	✓
TI	✓	✓	✓
Pairing Operations	3	2	1
Assumptions	DBDH, mDLIN	CBDH, CDH	q-ABDHE, CDH

Note: CBDH→Computational Bilinear Diffie-Hellman, CDH→Computational Diffie-Hellman, DBDH→Decisional Bilinear Diffie-Hellman, mDLIN→Modified Decisional Linear, q-ABDHE→Decisional Augmented Bilinear Diffie-Hellman Exponent

cost and time-consuming operations. This will be less favourable for implementation on lightweight devices with less resources of computing power and storage space. Pu et al. [15] have proposed a lightweight PAEKS scheme with MCI security model mainly designed for Industrial Internet of Things (IIoT) devices with limited resources by improving the algorithms in Type-1 pairing setting. This is achieved with precomputation of bilinear pairing operation $u = (g, g)$ during system initialisation and eliminates bilinear pairing operations in PAEKS and Trapdoor algorithms. Its overall running time is lesser than its pioneers, PAEKS [9] and PAEKS with MCI [16], as it only needs one pairing operation in the Test algorithm while retaining the security features. Table 1 summarises the comparison between Pu et al.'s PAEKS scheme and its predecessors, based on the supported security model and the total number of pairing operations involved in the scheme algorithms. In this paper, we describe Pu et al.'s lightweight PAEKS scheme in the Type-3 pairing setting which is more efficient than Type-1 pairing, because Type-1 pairing normally takes a longer time to execute and requires higher communication cost given the same security level with Type-3 pairing.

1.1 Organisation of the Paper

This section outlines the structure of this paper. Section 2 covers a literature review of some related existing PEKS and PAEKS schemes. Section 3 presents some preliminary concepts. Section 4 presents Pu et al.'s original lightweight PAEKS scheme and our modified version in Type-3 pairing. Section 5 provides a detailed performance analysis of the Type-1 and Type-3 pairing PAEKS schemes. Lastly, Section 6 wraps up the discussions of this paper and future work.

2 Related Work

Boneh et al. [4] first presented the PEKS scheme and was further enhanced by many researchers. Rhee et al. [18] introduced PEKS with a designated tester to prevent entities other than the specified server from running searching algorithms to prevent KGA. Chen et al. [6] introduced the idea of dual-server-based PEKS in case only one server has full searching capabilities. Park et al.'s [14] conjunctive keyword search with PEKS (PECK) scheme allows multiple keywords search in one search query. Huang and Li [9] proposed PAEKS scheme to prevent IKGA by adding a key pair for the sender during the generation of keyword ciphertext to serve as an authentication feature over the ciphertext.

Building upon the standard PAEKS scheme [9], various researchers have proposed improvements to address its limitations and enhance security features. Noroozi and Eslami [12] imrpoved Huang and Li's PAEKS scheme with multiuser settings where the privacy of each user should be maintained when several users have permission to search for ciphertext. Li et al. [11] introduced designated-server identity-based authenticated encryption with keyword search (dIBAEKS) which combines designated server testability with PAEKS of identity-based variant to prevent offline KGA designed in Type-1 and Type-3 pairing. Chenam and Ali [7] proposed a scheme named designated cloud server-based multi-user certificateless authenticated encryption with conjunctive keyword search (dmCLPAECKS) that eliminates the need for a secure channel to communicate between designated server and receiver with designated server and multi-user settings. This scheme also supports conjunctive keyword searches. Zhang et al. [23] proposed a new scheme, designated server certificateless deniably authenticated encryption with keyword search (dCLDAEKS) where the deniably authenticated encryption (DAE) technique enables the data sender to deny their involvement after the communication. Zhang et al. [22] designed an identity-based authorised searchable encryption scheme (IBASE) for electronic health information systems.

Besides, researchers have also focused on enhancing the security models of CI and TI, as well as enhancing overall performance efficiency of the scheme. Qin et al. [16] added MCI security model on PAEKS to capture chosen multi-ciphertext attack. Pan and Li [13] extended the work from Qin et al. to achieve Multi-Trapdoor Indistinguishability (MTI) security along with MCI to prevent adversaries from performing frequency analysis over the trapdoors with tuples of keywords. Qin et al. [17] improved the security of the MCI to fully MCI where an attacker can obtain ciphertext encrypted with any keyword including the challenge ones. Cheng et al. [8] also extended the TI security to fully MTI security, their scheme has stronger security by achieving both full MTI and full MCI to prevent fully chosen keyword to cipher-keyword attacks (FCKCA). Yang et al. [20] proposed secure-channel free public-key authenticated encryption with a multi-keyword search (SCF-PAEMKS) scheme, avoiding the requirement of a secure channel and supporting conjunctive keyword search along with MCI and MTI. Pu et al. [15] improved Qin et al.'s [16] scheme to have better performance that is suitable for lightweight devices in IIoT. Shiraly et al. [19] constructed a pairing-free certificateless PAEKS scheme to remove key management and perform better. Bai et al. [3] presented a pairing-free PAEKS scheme based on elliptic curves which also have an MCI security model. Yao et al.'s [21] PAEKS scheme is secure against chosen-ciphertext attack (CCA) based on lattice algorithms to prevent quantum attacks and scalable in practice to support conjunctive keyword search and multi-user settings.

3 Preliminaries

3.1 Bilinear Pairings

Most cryptographic schemes are developed with bilinear pairing which is defined as $e : G_1 \times G_2 \rightarrow G_T$ where G_1, G_2 and G_T are cyclic groups of prime order p. It carries the following properties:

1. Bilinearity. For any $g_1 \in G_1$, $g_2 \in G_2$, $a, b \in Z_p$, then $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. Non-degeneracy. $e(g_1, g_2) \neq 1$.
3. Computability. For any $g_1 \in G_1$, $g_2 \in G_2$, there is an efficient algorithm to compute $e(g_1, g_2)$.

The types of bilinear pairings:

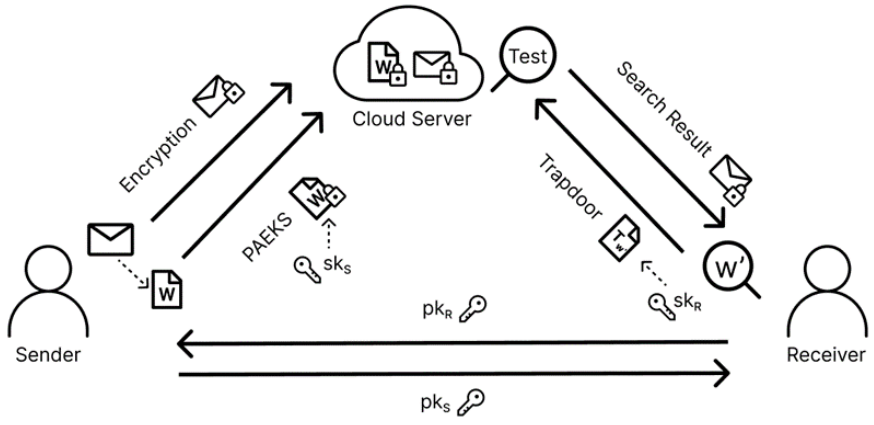


Figure 1: PAEKS Scheme.

- Type-1: $G_1 = G_2$.
- Type-2: $G_1 \neq G_2$, but an efficiently computable homomorphism from G_2 to G_1 exists.
- Type-3: $G_1 \neq G_2$, but no efficiently computable homomorphism from G_2 to G_1 exists.

3.2 Definition of PAEKS

In the standard PAEKS scheme illustrated in Figure 1, six algorithms are needed:

1. $Setup(\lambda) \rightarrow params$: With the input of a security parameter λ, it outputs a global system parameter $params$.
2. $KeyGen_S(params) \rightarrow (pk_S, sk_S)$: With the input of $params$, it outputs a pair of public key and secret key of sender (pk_S, sk_S).
3. $KeyGen_R(params) \rightarrow (pk_R, sk_R)$: With the input of $params$, it outputs a pair of public key and secret key of receiver (pk_R, sk_R).
4. $PAEKS(w, sk_S, pk_R) \rightarrow C_w$: With the input of a keyword w, secret key of sender sk_S, and public key of receiver pk_R, it encrypts w and outputs the corresponding ciphertext C_w.
5. $Trapdoor(w', pk_S, sk_R) \rightarrow T_{w'}$: With the input of a keyword w', public key of sender pk_S, and secret key of receiver sk_R, it generates the corresponding trapdoor of the keyword $T_{w'}$.
6. $Test(pk_S, pk_R, C_w, T_{w'}) \rightarrow 1/0$: With the input of public key of sender pk_S, public key of receiver pk_R, a keyword ciphertext C_w, and a trapdoor $T_{w'}$, it outputs 1 if the keyword contained in the ciphertext is matching with the keyword of the trapdoor, else it outputs 0.

4 Pu et al.'s Lightweight PAEKS Scheme

This section presents the original constructions of Pu et al.'s lightweight PAEKS scheme [15] in Type-1 bilinear pairing and our enhanced construction in Type-3 bilinear pairing.

4.1 Pu et al.'s Lightweight PAEKS Scheme in Type-1 Pairing

The original constructions of Pu et al.'s lightweight PAEKS scheme [15] in Type-1 bilinear pairing are as follows:

- $Setup(\lambda) \rightarrow params$: With the input of a security parameter λ, select p, a large prime number and set bilinear pairing function $e : G_1 \times G_1 \rightarrow G_T$, where G_1 and G_T are cyclic groups of order p. Select g, random generator of G_1 and compute $u = e(g, g)$. Set two hash functions $H_1 : \{0,1\}^* \rightarrow Z_p^*$, which takes input of variable-length binary strings and outputs a fixed-length hash value that maps to Z_p^*, and $H_2 : G_T \rightarrow \{0,1\}^{|p|}$, which accepts input element of G_T and outputs a binary string with a length equal to the bit length of p. The final output is a global public parameter, $params = (G_1, G_T, p, e, g, u)$.
- $KeyGen_S(params) \rightarrow (pk_S, sk_S)$: With the input of $params$, randomly select $y \in Z_p^*$ and set sender's key pair $(pk_S, sk_S) = (g^y, y)$.
- $KeyGen_R(params) \rightarrow (pk_R, sk_R)$: With the input of $params$, randomly select $x \in Z_p^*$ and set receiver's key pair $(pk_R, sk_R) = (g^x, x)$.
- $PAEKS(params, w, sk_S, pk_R) \rightarrow C_w$: With the input of a keyword w, sk_S, and pk_R, randomly select $r \in Z_p^*$, then, compute for $A = H_2(u^{yr}), v = H_1(w||pk_R^y)$, and $B = (g^{vr} \cdot pk_R^r)$. The output is a keyword ciphertext $C_w = (A, B)$.
- $Trapdoor(params, w', pk_S, sk_R) \rightarrow T_{w'}$: With the input of a keyword w', pk_S, and sk_R, compute for $v' = H_1(w'||pk_S^x)$ and $T_{w'} = pk_S^{\frac{1}{x+v'}}$, and output the trapdoor $T_{w'}$.
- $Test(params, C_w, T_{w'}) \rightarrow 1/0$: With the input of C_w and $T_{w'}$, compare $H_2(e(T_{w'}, B)) = A$. If the comparison is true, output 1; if false, output 0.

4.2 Pu et al.'s Lightweight PAEKS Scheme in Type-3 Pairing

We reconstruct the lightweight PAEKS [15] using Type-3 pairing below:

- $Setup(\lambda) \rightarrow params$: With the input of a security parameter λ, select p, a large prime number and set bilinear pairing function $e : G_1 \times G_2 \rightarrow G_T$, where G_1, G_2 and G_T are cyclic groups of order p. Select g_1, random generator of G_1, g_2, random generator of G_2 and compute $u = e(g_1, g_2)$. Set two hash functions $H_1 : \{0,1\}^* \rightarrow Z_p^*$, which takes input of variable-length binary strings and outputs a fixed-length hash value that maps to Z_p^*, and $H_2 : G_T \rightarrow \{0,1\}^{|p|}$, which accepts input element of G_T and outputs a binary string with a length equal to the bit length of p. The final output is a global public parameter, $params = (G_1, G_2, G_T, p, e, g_1, g_2, u)$.
- $KeyGen_S(params) \rightarrow (pk_{S1}, pk_{S2}, sk_S)$: With the input of $params$, randomly select $y \in Z_p^*$ and set sender's key pair $(pk_{S1}, pk_{S2}, sk_S) = (g_1^y, g_2^y, y)$.
- $KeyGen_R(params) \rightarrow (pk_R, sk_R)$: With the input of $params$, randomly select $x \in Z_p^*$ and set receiver's key pair $(pk_R, sk_R) = (g_1^x, x)$.
- $PAEKS(params, w, sk_S, pk_R) \rightarrow C_w$: With the input of a keyword w, sk_S, and pk_R, randomly select $r \in Z_p^*$, then, compute for $A = H_2(u^{yr}), v = H_1(w||pk_R^y)$, and $B = (g_1^{vr} \cdot pk_R^r)$. The output is a keyword ciphertext $C_w = (A, B)$.
- $Trapdoor(params, w', pk_{S1}, pk_{S2}, sk_R) \rightarrow T_{w'}$: With the input of a keyword w', pk_{S1}, pk_{S2}, and sk_R, compute for $v' = H_1(w'||pk_{S1}^x)$ and $T_{w'} = pk_{S2}^{\frac{1}{x+v'}}$, and output the trapdoor $T_{w'}$.

- $Test(params, C_w, T_{w'}) \rightarrow 1/0$: With the input of C_w and $T_{w'}$, compare $H_2(e(T_{w'}, B)) = A$. If the comparison is true, output 1; if false, output 0.

In the Test step, the correctness of Type-3 lightweight PAEKS scheme is verified as follows:

$$
\begin{aligned}
& H_2(e(T_{w'}, B)) \\
& = H_2(e(pk_{S2}^{\frac{1}{x+v'}}, g_1^{vr} \cdot pk_R^r)) \\
& = H_2(e(g_2^{y \cdot \frac{1}{x+v'}}, g_1^{vr} \cdot g_1^{xr})) \\
& = H_2(e(g_2^{\frac{y}{x+v'}}, g_1^{r(x+v)})) \\
& = H_2(e(g_1, g_2)^{\frac{yr(x+v)}{x+v'}}) \\
& = H_2(e(g_1, g_2)^{yr}) \\
& A = H_2(u^y r) = H_2(e(g_1, g_2)^{yr})
\end{aligned}
$$

In our enhanced version, the sender has two public keys generated from different groups: pk_{S1} from G_1 and pk_{S2} from G_2. This is to satisfy $v = v'$ for the correctness of the Test algorithm. Notably, G_2 elements have a larger size compared to G_1 elements due to G_2 is a subgroup of the elliptic curve with a similar equation to G_1 but with a different embedding degree [10]. To minimise the storage space needed for ciphertexts in the database, the algorithms are designed to generate a smaller-sized keyword ciphertext using G_1, which is stored permanently in the database, and a larger-sized trapdoor using G_2, which is generated for temporary use during the search.

4.3 Security Analysis

Theorem 1. *Based on the decisional q-ABDHE and CDH assumptions, the modified PAEKS scheme in Type-3 pairing can resist IKGA by achieving both MCI and trapdoor privacy.*

The original Pu et al.'s lightweight PAEKS scheme in Type-1 pairing is proven secure against IKGA using decisional q-ABDHE and CDH assumptions. When a cryptographic scheme is transformed from Type-1 to Type-3 bilinear pairing, its underlying secruity assumptions are also transformed. Existing works [2] have proven that if the original assumptions are valid in the Type-1 generic bilinear group model, they are also valid in the Type-3 model. Specifically, the underlying computational difficulty is extended from the symmetric group in Type-1 pairing to the asymmetric groups in Type-3 pairing [2]. Thus, the security of the scheme in the Type-3 setting follows from its security in the Type-1 setting.

5 Performance Analysis

In this section, the performance of Pu et al.'s lightweight PAEKS is compared between Type-1 and Type-3 pairing settings based on algorithm execution time and memory usage. The lightweight PAEKS scheme is implemented using Python Charm-Crypto library, which is built on the Pairing-Based Cryptography (PBC) library. The performance analysis is conducted on a laptop with an AMD Ryzen 2.10 GHz processor and 4GB RAM, running Kali Linux 2022 in a virtual machine. The asymmetric pairing curve used for Type-3 setting is BN254, which has approximately 90-bit security level. For the Type-1 setting, the symmetric curve SS1024 is used, which provides 50-bit security level according to the SafeCurve website [1]. The source code is available on GitHub.

Table 2 and Figure 2 show the time taken to execute each algorithm in milliseconds, including setup, sender key generation, receiver key generation, PAEKS encryption, trapdoor generation and test. Overall, the results show that Type-3 PAEKS requires a shorter time compared to Type-1 PAEKS. Each algorithm of Type-3 PAEKS executes faster than Type-1 PAEKS except for setup and test algorithms. To ensure a fair comparison between different pairing settings of the scheme, pairing curves with similar security levels should be chosen. However, due to the limited options of curves provided by the Charm-Crypto library, BN254 and SS1024 are the most suitable pair of curves available. Despite their differing security levels, the analysis results demonstrate that Type-3 scheme of 90-bit security outperforms Type-1 scheme even when Type-1 is at a lower security level of 50-bit. Given that execution time grows exponentially with the security level, Type-3 scheme remains faster even if a Type-1 curve with a larger base field, such as 2048-bit, were used to match the security level of Type-3.

Table 2: Type-1 vs Type-3 PAEKS Algorithms Execution Time.

Scheme	Time Taken (ms)					
	$Setup$	$KeyGen_S$	$KeyGen_R$	$PAEKS$	$Trapdoor$	$Test$
Type-1	21.3	15.4	15.7	53.3	32.0	21.8
Type-3	28.2	2.9	1.3	13.6	3.1	24.2

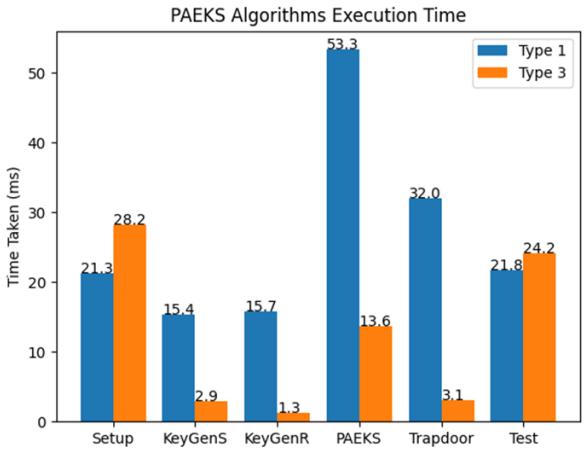


Figure 2: Type-1 vs Type-3 PAEKS Algorithms Execution Time.

Table 3: Type-1 vs Type-3 Pairing Curve Parameter Sizes.

Curve	Size (bits)				
	Prime (q)	Order (p)	Z_p	G_1	G_2
Type-1 SS1024	1033	1024	1024	2066	2066
Type-3 BN254	254	254	254	508	1016

Table 3 illustrates the parameter sizes used in Type-1 SS1024 and Type-3 BN254

pairing curves. The private key size depends on the order, denoted as p. The public key is determined by the sizes of G_1 and G_2, which are based on the prime, q size. G_1 and G_2 of Type-1 pairings do not differ in size, both are twice the size of q. In contrast, G_1 of Type-3 pairings is twice the size of q, while G_2 is four times the size of q. Consequently, the total size of the two sender public keys, one from G_1 and the other from G_2, is the sum of the sizes of both groups. The receiver's public key size is based on G_1. The ciphertext size is calculated by adding the sizes of G_1 and Z_p. The trapdoor of Type-1 is the same size as G_1, while Type-3 trapdoor size is the same as G_2. Table 4 and Figure 3 present the communication cost measured in bits for the private key, sender public key, receiver public key, keyword ciphertext and trapdoor. Type-3 pairing not only provides better security but also requires less communication overhead compared to Type-1 pairing.

Table 4: Type-1 vs Type-3 Communication Cost.

Scheme	Size (bits)						
	sk	pk_S			pk_R	C_w	T_w
Type-1	1024	2066			2066	3090	2066
Type-3	254	pk_{S1}	pk_{S2}	Total	508	762	1016
		508	1016	1524			

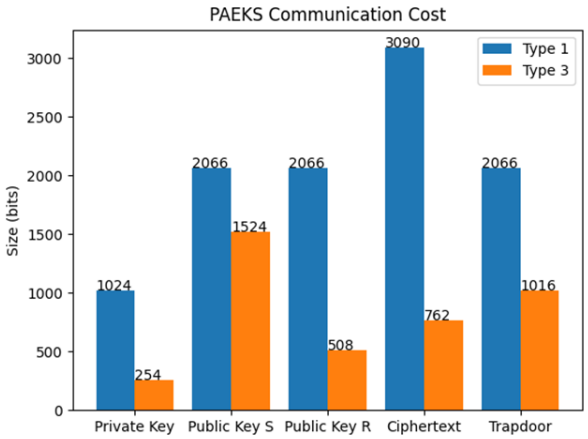


Figure 3: Type-1 vs Type-3 Communication Cost.

6 Conclusion

In conclusion, Pu et al.'s lightweight PAEKS scheme can support the searching of data in encrypted form with additional privacy-preserving features, including allowing only authenticated users to generate valid keyword ciphertext and MCI security to avoid KGA. With minimal bilinear pairing operation required in the scheme, the scheme is user-friendly to be implemented for lightweight devices which require efficient computation. We have modified the scheme to a Type-3 pairing setting to further enhance its performance efficiency and analysed its implementation performance. To further enhance the security features of the PAEKS scheme, incorporating the MTI security model can be beneficial

to prevent attacks that attempt to differentiate trapdoors containing the same keywords from different sets of keywords. Additionally, the scheme could be further enhanced by incorporating features such as conjunctive keyword search and fuzzy keyword search to efficiently perform multiple keyword searches or fuzzy queries within single searches and minimise communication overhead, especially when searching through large datasets.

Acknowledgments

This work was supported by the Telekom Malaysia Research & Development Grant (RDTC/221045).

References

[1] SafeCurves: Introduction. https://safecurves.cr.yp.to/.

[2] Masayuki Abe, Fumitaka Hoshino, and Miyako Ohkubo. Design in type-i, run in type-iii: Fast and scalable bilinear-type conversion using integer programming. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 387–415. Springer, 2016.

[3] Lisha Bai, Lei Yong, Zhixian Chen, and Jun Shao. Pairing-free public-key authenticated encryption with keyword search. *Comput. Stand. Interfaces*, 88:103793, 2024.

[4] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

[5] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, volume 4165 of *Lecture Notes in Computer Science*, pages 75–83. Springer, 2006.

[6] Rongmao Chen, Yi Mu, Guomin Yang, Fuchun Guo, and Xiaofen Wang. Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE Trans. Inf. Forensics Secur.*, 11(4):789–798, 2016.

[7] Venkata Bhikshapathi Chenam and Syed Taqi Ali. A designated cloud server-based multi-user certificateless public key authenticated encryption with conjunctive keyword search against IKGA. *Comput. Stand. Interfaces*, 81:103603, 2022.

[8] Leixiao Cheng, Jing Qin, Feng Feng, and Fei Meng. Security-enhanced public-key authenticated searchable encryption. *Information Sciences*, 647:119454, November 2023.

[9] Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Cryptology ePrint Archive, Report 2018/007, 2018. https://eprint.iacr.org/2018/007.

[10] Tetsuya Izuta, Yasuyuki Nogami, and Yoshitaka Morikawa. Efficient non symmetric pairing groups on ordinary pairing friendly curve of embedding degree 1. In *Proceedings of IEEE International Conference on Communications, ICC 2011, Kyoto, Japan, 5-9 June, 2011*, pages 1–5. IEEE, 2011.

[11] Hongbo Li, Qiong Huang, Jian Shen, Guomin Yang, and Willy Susilo. Designated-server identity-based authenticated encryption with keyword search for encrypted emails. *Inf. Sci.*, 481:330–343, 2019.

[12] Mahnaz Noroozi and Ziba Eslami. Public key authenticated encryption with keyword search: revisited. *IET Inf. Secur.*, 13(4):336–342, 2019.

[13] Xiangyu Pan and Fagen Li. Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *J. Syst. Archit.*, 115:102075, 2021.

[14] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public key encryption with conjunctive field keyword search. In Chae Hoon Lim and Moti Yung, editors, *WISA 04*, volume 3325 of *LNCS*, pages 73–86. Springer, Heidelberg, August 2004.

[15] Lang Pu, Chao Lin, Biwen Chen, and Debiao He. User-friendly public-key authenticated encryption with keyword search for industrial internet of things. *IEEE Internet Things J.*, 10(15):13544–13555, 2023.

[16] Baodong Qin, Yu Chen, Qiong Huang, Ximeng Liu, and Dong Zheng. Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Inf. Sci.*, 516:515–528, 2020.

[17] Baodong Qin, Hui Cui, Xiaokun Zheng, and Dong Zheng. Improved security model for public-key authenticated encryption with keyword search. In Qiong Huang and Yu Yu, editors, *Provable and Practical Security - 15th International Conference, ProvSec 2021, Guangzhou, China, November 5-8, 2021, Proceedings*, volume 13059 of *Lecture Notes in Computer Science*, pages 19–38. Springer, 2021.

[18] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.*, 83(5):763–771, 2010.

[19] Danial Shiraly, Nasrollah Pakniat, Mahnaz Noroozi, and Ziba Eslami. Pairing-free certificateless authenticated encryption with keyword search. *J. Syst. Archit.*, 124:102390, 2022.

[20] Pan Yang, Hongbo Li, Jianye Huang, Hao Zhang, Man Ho Allen Au, and Qiong Huang. Secure channel free public key authenticated encryption with multi-keyword search on healthcare systems. *Future Gener. Comput. Syst.*, 145:511–520, 2023.

[21] Lisha Yao, Jian Weng, Anjia Yang, Xiaojian Liang, Zhenghao Wu, Zike Jiang, and Lin Hou. Scalable cca-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing. *Inf. Sci.*, 624:777–795, 2023.

[22] Xiaojun Zhang, Chunxiang Xu, Liming Mu, and Jie Zhao. Identity-based encryption with keyword search from lattice assumption. *China Communications*, 15:164–178, 04 2018.

[23] Yulei Zhang, Long Wen, Yongjie Zhang, and Caifen Wang. Designated server certificateless deniably authenticated encryption with keyword search. *IEEE Access*, 7:146542–146551, 2019.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 188–194.

Securing Nation's Digital Future: A Proposed Transition to Post-Quantum Cryptography

Mohamad Fariq Burhan, Hilmy Nawawi and Muhammad Rezal Kamel Ariffin[1]

Department of Mathematics and Statistics, Faculty of Science and Institute for Mathematical Research, Universiti Putra Malaysia {rezal}@upm.edu.my

Abstract.
The rise of quantum computing will significantly threaten the security of conventional cryptographic algorithms, jeopardizing the confidentiality and integrity of sensitive data. This literature explores the challenges posed by quantum computers and proposes a roadmap for transitioning government services and applications to post-quantum cryptography (PQC). By proactively embracing PQC, the government can better safeguard her critical information and possibly ensure a vibrant and resilient landscape for a secure digital future. The paper also presents a feasibility study involving industries developing PQC as an experimental product. The immediate objective during the transition period is to choose the best implementation strategy, with a current strategic focus on deploying hybrid solutions that integrate PQC techniques with classical methods. Until global consensus is achieved on PQC algorithm adoption, national PQC strategies must be part of hybrid solutions to ensure high crypto-agility.

Keywords: post quantum cryptography (PQC) · PQC hybrid algorithm · crypto-agility

1 Introduction

The digital revolution has undeniably woven itself into the fabric of our daily lives, influencing everything from communication and commerce to governance and national security. At the bottom of this digital world is cryptography, the science of safeguarding information using complex mathematical algorithms [9]. These algorithms play vital roles in ensuring confidentiality, integrity, authenticity, and non-repudiation of data and communications [15], and they also work together to protect sensitive information, financial transactions, personal records, and government communications, among others.

However, the ecosystem is going through alarming changes due to the advent of applied quantum mechanical technology, especially in its computational power segment. Quantum technology can utilize qubits' remarkable ability to exist in a superposition of states, meaning they can represent both 0 and 1 simultaneously [14]. This unique capability allows quantum computers to solve certain mathematical problems and perform complex calculations significantly faster than classical computers, particularly for tasks that leverage quantum algorithms for factoring large numbers. Thus, it poses a daunting risk to widely used asymmetrical cryptographic algorithms like RSA and ECC, potentially rendering them vulnerable to attacks by powerful quantum computers in the future [5].

International communities are competing to develop a cryptanalytically relevant quantum computer (CRQC), as the mathematical proof of [17] breaches various current cryptography standards. In November 2022, IBM launched its 433-qubit "Osprey" processor, which was the world's fastest quantum computer at the time, before Atom Computing

unveiled a 1000-qubit processor in October 2023. Subsequently, IBM made headlines again in December 2023 with the Condor, featuring 1,121 superconducting qubits. China officially entered the race in January 2024 with the introduction of Origin Wukong, its first homegrown, third-generation superconducting quantum computer, which is now available for global users to trial.

Moving to PQC is not just a technical exercise; it is a necessary step to protect sensitive national data and keep critical infrastructure safe. ENISA underscores this necessity, highlighting the potential economic and national security risks associated with failure to prepare for the quantum threat [6]. A report from the National Academies of Sciences, Engineering, and Medicine (NASEM) further underscores the urgency of adopting PQC, emphasizing its critical role in safeguarding sensitive information in sectors such as healthcare, finance, and national security. To address these risks, PQC standards must be established. The National Institute of Standards and Technology (NIST) has been leading this effort, with various algorithms under consideration for standardisation.

2 Post Quantum Cryptography

The establishment of post-quantum cryptography standards is one of the mechanisms used to mitigate the challenges posed by quantum computation. Consequently, the National Institute of Standards and Technology (NIST) in the United States launched a global initiative in February 2016 to invite contributions from scientists and mathematicians worldwide. As of the end of November 2017, NIST had received a total of 82 draft algorithms. After initial screening processes, NIST published 69 drafts, with algorithms derived from five main mathematical methods: Lattices [13] [3], Codes [12], Multivariates [11], and Hashes for signatures [18].

Following that, NIST released four shortlisted potential quantum-resistant cryptographic algorithms [10] [2], vying for the acclaimed standardization in July 2022. `The Crystals-Kyber` algorithm was designated for the key encapsulation mechanism that consists of symmetrical session keys enveloped in the recipient's public key due to its smaller footprint compared to other contenders. This translates into easier facilitation of key exchange as well as a higher speed of operation. `The Crystals-Dilithium, Falcon, and Sphinc+` algorithms were reviewed as particularly efficient for digital signatures as an additional element in key exchanging protocol TLS. Subsequently, NIST released four shortlisted potential.

Nonetheless, NIST announced in July 2023 that the PQC standardisation process will continue with a fourth round, with BIKE, Classic McEliece, HQC, and SIKE encapsulation mechanisms still under consideration.

2.1 Hybrid Mode Approach: The Google Chrome Model

Presently, organisations seeking quantum-resistant security are likely to gravitate towards hybrid cryptographic solutions that integrate both classical and post-quantum algorithms. The adoption of post-quantum cryptography allows organisations to strengthen their existing security infrastructure with a future-proof approach by ensuring the continued effectiveness of current security measures while adding a layer of protection against potential threats from quantum computers. This hybrid strategy has a significant number of advocacies from the World Economic Forum [4], the National Cyber Security Centre (NCSC) of the U.K. [20], the Federal Office for Information Security (BSI) of Germany, the European Union Agency for Cybersecurity, the French Network and Information Security Agency, and the White House of the U.S. [16].

On August 15, 2023, Google Chrome version 116 released a quantum hybrid key agreement mechanism, providing users with meaningful opportunities to engage in PQC

for HTTPS [15]. Verification from the antropol.com website indicates that the Google.com servers are currently using the X25519 Kyber-768 hybrid key exchange algorithm on TLS version 1.3. The approach advances data security by employing a robust combination of cryptographic techniques.

`Unique Symmetric Keys for Each Session:` To provide optimal protection, a new, temporary symmetric key, X25519, is generated for each communication session. This approach eliminates the risk of compromising previous or future sessions if a key were to be exposed.

`Quantum-Resistant Key Exchange:` This innovative PQC mechanism, Kyber-768, safeguards the exchange of the symmetric key. It encrypts the key using the server's public key, ensuring only the server can unlock it with its private key. If an attacker intercepts it, they cannot decrypt it without the server's private key, providing exceptional security.

`Separate Encryption for Data:` Once the secure key exchange is complete, a separate, well-established symmetric encryption algorithm like AES takes over and encrypts the actual data, providing robust confidentiality.

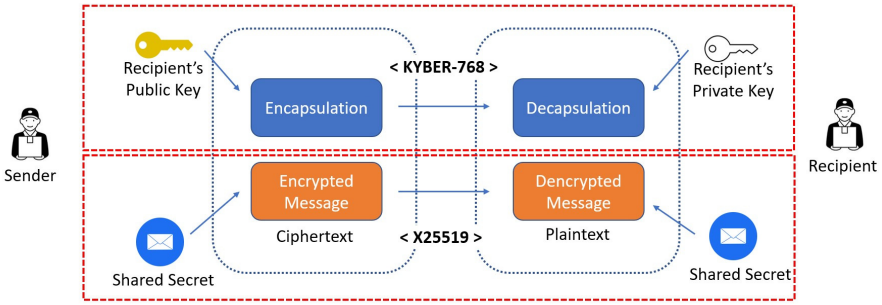


Figure 1: This illustration demonstrates how the X25519Kyber768 key encryption method (KEM) works to encapsulate and decapsulate symmetric key data.

While the Kyber algorithm remains under development and standardization is yet to come, its inclusion in browsers like Chrome represents a significant step forward. With the move, Google has served as a beacon for other companies, potentially accelerating industry-wide adoption of PQC. By taking a proactive stance, Google is also fostering a collaborative environment that will ultimately benefit everyone as PQC becomes the new standard and Kyber's early implementation paves the way for greater interoperability between different systems.

There exists an alternative method for utilising the post-quantum hybrid algorithm that extends beyond the conventional approach. This method involves utilising either the Chrome Developer Tools or the Cloudflare Research PQC Key Agreement Checking Tool, which requires manual activation due to its current experimental status. The X25519Kyber-768 client-side support in Chrome version 116 can then be enabled by following these steps: Open Google Chrome and launch a new tab. Type "chrome://flags" in the address bar and press Enter (navigate the Chrome Experiments page). Type "TLS 1.3 hybridized Kyber support" within the search bar and press Enter. Change status form “default” to “enabled on the setting label. Click “Relaunch” button to complete the process.

2.2 Public Services as The Prime Mover

Transitioning to PQC involves a well-defined roadmap that leverages best practices from various countries while considering developing countries' specific needs and resources. A comprehensive plan for developing the master plan should include risk assessment and

prioritization, standardization, and pilot projects, as well as training and awareness campaigns. Additionally, it must address gradual deployment, cost considerations, cooperation, and knowledge sharing. Released on July 12, 2023, the South Korea master plan mandates the federal government to fully integrate PQC technology into South Korea's national encryption systems by 2035. By 2024, the government must construct actionable PQC integration plans for acquiring the technology, changing legislation, building industrial bases, and advancing assurance infrastructure and cryptographic transformation to achieve this target.

Countries with limited resources and expertise could adopt a similar hybrid transitioning approach, currently being implemented by some developed nations and multinational corporations. This approach could begin with several non-essential services, such as the distribution of essential staples, vehicles, and online applications for business registration and licensing, as part of a multi-stage pilot initiative before large-scale deployment [8]. Through these pilots, each country could test and evaluate their PQC algorithms alongside current ones to assess their performances in terms of efficiency, speed, and resilience, so that, in the end, they would be able to develop tailored solutions for each individual nation.

One of the advantages of a hybrid approach to data security is its ability to protect data from both current threats and those emerging from the quantum computer era, by providing a fail-safe mechanism in the event of a compromised data transmission element. Google Chrome is implementing a hybrid scheme that continuously protects connections using existing key exchange algorithms, while also allowing for the deployment and testing of new quantum-resistant algorithms. The method plays a crucial role in safeguarding against the "harvest now, decrypt later" (HNDL) cryptanalysis exploit, which continuously captures and stores encrypted data until more powerful quantum technology becomes available.

Conducting a thorough risk assessment, similar to the Germany BSI PQC plan, will be essential throughout this first stage. According to the valuation, state authorities should determine their critical data assets, identify quantum attack vulnerabilities in data and transmission, and consider possible transitional interruptions. The US government's approach [1] suggests prioritizing high-impact government services that manage sensitive data at the next level, which could involve several stages; E-government services, Government procurement application platforms, The process involves registering and identifying births and deaths, and Healthcare management systems.

2.3 Crypto Agility – Initiating Ease of Modification Procedures Upon Deployed PQC and the Advantages of Utilizing Secure Locally Developed PQC

Experts from various countries can eventually adjust certain aspects of the published PQC algorithms by conducting continuous risk assessments during the pilot phase, leading to the integration and development of a homegrown PQC product. This is part of the PQC standardization project, actively promoted by NIST [10], the Chinese Academy of Sciences (CAS), and the European Telecommunications Standards Institute (ETSI).

In the "Report on Post-Quantum Cryptography", NIST points out right from the beginning that it "recognises the challenge of moving to new cryptographic infrastructures and therefore emphasises the need for agencies to focus on crypto agility." Crypto-agility is the backbone of the hybrid approach [7]. In simple terms, agility is defined as a system's ability to adapt quickly to new approaches. Crypto-agility not only encourages system development and evolution but also acts as a safety measure or incident response mechanism. Thus, to ensure ease of agility, we put forward the idea that countries are advised to create their own homegrown PQC algorithms based on the guidelines provided by organisations like NIST in order to better complement the crypto-agility approach and

emphasise the security component.

We hypothesise that the national cryptographic community would better understand locally developed PQC algorithms, ensuring a high agility rate when upgrading and/or modification is required. Current developments via the NIST PQC forum suggest that the initial four PQC selected by NIST are still in need of in-depth security analysis. As such, combining locally PQC techniques with "internationally adopted" PQC algorithms would provide an extra layer of cryptographic security that is highly modifiable. Our proposed strategy is practical for national digital sovereignty until we reach a global consensus on PQC algorithm adoption. A proposed overview of the PQC improved with homegrown PQC to support Crypto-Agility in Google Chrome is provided in Figure 2.

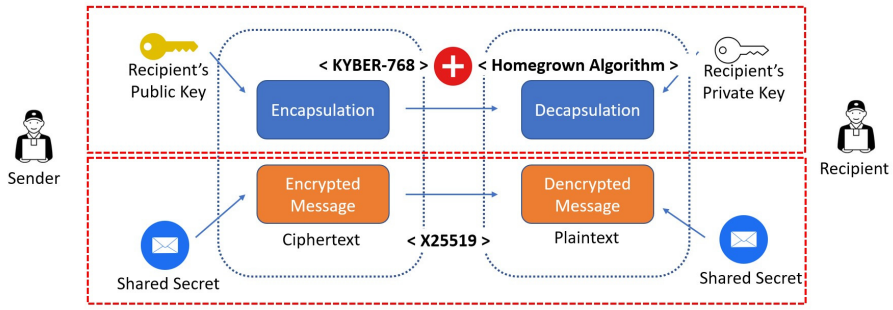


Figure 2: An overview of the PQC improved with homegrown PQC to support Crypto-Agility in Google Chrome.

Consequently, we must optimise PQC to meet the performance requirements of existing applications, which may include resource-constrained environments. Simultaneously, we must continuously facilitate the development of standardised integration protocols to enable interoperability between PQC and existing systems.

Collaboration and knowledge sharing with regional and international partners fosters collective resilience against the quantum threat, as highly emphasized in the Quantum-Safe Cryptography Roadmap, developed by the Global Forum on Cyber Expertise (GFCE) on the importance of international cooperation in research, development, and deployment of PQC solutions. In the regional spectrum, initiatives like the ASEAN Cybersecurity Cooperation Framework (ACCF) can serve as valuable platforms for knowledge exchange and capacity building related to PQC adoption.

3 Recommendations for Malaysia

In order to effectively transition to PQC and manage associated costs, Malaysia should adopt a multifaceted approach involving the government, academic institutions, and the corporate sector. Our objective is to provide Malaysia with the following recommendations to assist in the implementation of its PQC strategy:

`Quantum-Resistant Key Exchange:`Hybrid Approach Adoption: Malaysia should begin integrating PQC with its existing classical cryptographic systems. This hybrid model offers immediate protection and allows for the evaluation and fine-tuning of PQC techniques. Starting with critical government and public sector applications will lay a strong foundation for broader adoption.

`National PQC Strategy Development:` To ensure a successful transition to PQC, a comprehensive national strategy is required [39]. Key components of this strategy should encompass the implementation of pilot projects, thorough risk assessments, and the development of standardised PQC integration protocols. International and regional

cybersecurity collaboration is essential for generating valuable insights and supporting the creation of robust PQC solutions.

`Research and Development:` Investing in local Research and Development efforts to develop and refine PQC algorithms is critical. Encouraging collaboration between academic institutions, government agencies, and industry will spur innovation and ensure that Malaysia's PQC solutions are both effective and adaptable. This collaboration should extend to joint research initiatives, workshops, and technology exchange programs with international partners.

`Cybersecurity Education Enhancement:` Training and awareness programs are vital for equipping personnel with the knowledge and skills needed for PQC implementation. Malaysia should prioritise education initiatives to build a skilled workforce capable of managing and advancing PQC solutions.

`Open-Source Library Utilization:` To optimize costs and ensure effective implementation, Malaysia should leverage open-source PQC libraries. This approach will enable cost-effective deployment and adaptation of PQC technologies while promoting transparency and collaboration within the global cybersecurity community.

`Fostering Regional and International Collaboration:` Effective collaboration with regional and international counterparts is indispensable for advancing knowledge and capabilities in PQC. Malaysia's active participation in global platforms, such as those convened by the Global Forum on Cyber Expertise (GFCE), is essential for staying informed about PQC advancements and contributing to global cybersecurity initiatives.

4 Conclusion and Future Work

Quantum computing's impending threat underscores the criticality of a timely and comprehensive transition to PQC. This research highlights the imperative for a collaborative, multi-stakeholder approach involving government, academia, and industry to effectively implement PQC. By investing in research, development, and standardisation, Malaysia can position itself at the forefront of quantum-resistant cybersecurity. A proactive stance coupled with international cooperation is essential to mitigate the risks associated with quantum computing and secure a resilient digital future.

Although this research provides a solid foundation for understanding the challenges and opportunities associated with the transition to PQC, additional investigation is necessary to fully realise its full potential. Conducting a detailed cost-benefit analysis is essential to assessing the economic impact of both adopting and postponing PQC implementation. Moreover, thorough performance evaluations of various PQC algorithms across diverse hardware platforms are necessary to identify the most effective solutions. Investigating public perception and acceptance of PQC through user acceptance studies will be crucial for facilitating widespread understanding and adoption. Additionally, enhancing quantum threat modelling to identify specific vulnerabilities will help prioritise effective mitigation strategies.

References

[1] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the first round of the nist post-quantum cryptography standardization process. 2019.

[2] Gorjan Alagic, Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Yi-Kai Liu, Carl Miller, et al. Status report on the third round of the nist post-quantum cryptography standardization process. 2022.

[3] Rameez Asif. Post-quantum cryptosystems for internet-of-things: A survey on lattice-based algorithms. *IoT*, 2(1):71–91, 2021.

[4] Filipe Beato, Anne Ardon, Itan Barmes, Chris Knackstedt, et al. Transitioning to a quantum-secure economy. World Economic Forum, 2022.

[5] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[6] Ward Beullens, Jan-Pieter D'Anvers, Andreas T Hülsing, Tanja Lange, Lorenz Panny, Cyprien de Saint Guilhem, and Nigel P Smart. Post-quantum cryptography: Current state and quantum mitigation. 2021.

[7] Lily Chen, Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray A Perlner, and Daniel Smith-Tone. *Report on post-quantum cryptography*, volume 12. US Department of Commerce, National Institute of Standards and Technology . . . , 2016.

[8] Haohan Ding, Jiawei Tian, Wei Yu, David I Wilson, Brent R Young, Xiaohui Cui, Xing Xin, Zhenyu Wang, and Wei Li. The application of artificial intelligence and big data in the food industry. *Foods*, 12(24):4511, 2023.

[9] William Stallings Fifth Edition. Cryptography and network security, 2023.

[10] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4*, pages 19–34. Springer, 2011.

[11] Randy Kuang, Maria Perepechaenko, and Michel Barbeau. A new post-quantum multivariate polynomial public key encapsulation algorithm. *Quantum Information Processing*, 21(10):360, 2022.

[12] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[13] Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. Post-quantum lattice-based cryptography implementations: A survey. *ACM Computing Surveys (CSUR)*, 51(6):1–41, 2019.

[14] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*, volume 2. Cambridge university press Cambridge, 2001.

[15] D O'Brien. Protecting chrome traffic with hybrid kyber kem, 2023.

[16] BRIEFING ROOM. National security memorandum on promoting united states leadership in quantum computing while mitigating risks to vulnerable cryptographic systems. 2022.

[17] 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.

[18] HL Yin, Y Fu, CL Li, CX Weng, BH Li, J Gu, YS Lu, S Huang, and ZB Chen. Experimental quantum secure network with digital signatures and encryption. arxiv 2021. *arXiv preprint arXiv:2107.14089*.

Proceedings of the 9th International Cryptology and Information Security Conference 2024
CRYPTOLOGY2024, pp. 195–209.

MySEAL: A National Trusted Cryptographic Algorithm List

Nurul Amiera Sakinah Abdul Jamal[1], Yen Yee Chan[1,2], Nik Azura Nik Abdullah[1] and Hazlin Abdul Rani[1]

[1] CyberSecurity Malaysia, Selangor, Malaysia
{researcher.amiera,azura,hazlin}@cybersecurity.my
[2] Universiti Sains Malaysia, Penang, malaysia chan.yen.yee.cyy@gmail.com

Abstract. National security relies not only on cryptographic algorithms but also on their robustness in ensuring secure communication and data protection. Recognizing this critical aspect, Malaysia initiated a project known as the National Trusted Cryptographic Algorithm List (MySEAL) to compile a catalog of trusted cryptographic algorithms for national utilisation. This paper delves into two parts of MySEAL: AKSA MySEAL (prevalent cryptographic algorithms selected from various standards) and AKBA MySEAL (novel algorithms or published algorithms excluded in existing standards or other cryptographic algorithm listing projects). Candidates for AKSA MySEAL and AKBA MySEAL were comprehensively evaluated by cryptographic experts using a newly developed scoring matrix. The MySEAL project encompasses six cryptographic primitives, and, as such, this paper outlines the submission and evaluation criteria for AKSA MySEAL and AKBA MySEAL for each of these primitives. MySEAL submission and evaluation criteria cover the security and performance aspects of the candidate algorithms. Finally, this paper presents a list of the approved algorithms included in AKSA MySEAL, along with a brief overview of the ongoing developments in AKSA MySEAL 2.0 and AKSA MySEAL 2.1.

Keywords: MySEAL · AKSA MySEAL · AKBA MySEAL · CyberSecurity Malaysia · trusted cryptographic algorithm · cryptographic standard

1 Introduction

The National Trusted Cryptographic Algorithm List (MySEAL) is a national initiative aimed at creating a collection of reliable cryptographic algorithms [22]. This initiative focuses on developing a portfolio of cryptographic algorithms that are well-suited for implementation within the Malaysian context and align with the National Cryptography Policy (NCP). The NCP serves as a guiding document for Malaysia's pursuit of cryptographic sovereignty, and MySEAL contributes to the scientific aspects of cryptography and cryptanalysis. Furthermore, the objective of MySEAL is to serve as a comprehensive resource for the implementation of cryptographic algorithms in information security systems, providing valuable guidance and references for users. MySEAL aims to enhance the protection and security of electronically communicated classified information in Malaysia between Government-to-Government (G2G), Government-to-Business (G2B), Business-to-Business (B2B) and to Malaysia's CNII agencies and organisations.

MySEAL was initiated in 2016 and consists of two categories of cryptographic algorithms: Existing Cryptographic Algorithms, also known as Algoritma Kriptografi Sedia Ada (AKSA) [24], and New Cryptographic Algorithms, also known as Algoritma Kriptografi Baharu (AKBA) [23]. AKSA refers to cryptographic algorithms that have already been published in recognised standards or have undergone thorough evaluation in established cryptographic

algorithm projects. AKSA includes algorithms endorsed by reputable organisations such as FIPS (Federal Information Processing Standards) [26], CRYPTREC (Cryptography Research and Evaluation Committees) [19], NESSIE (New European Schemes for Signatures, Integrity, and Encryption) [27], and eSTREAM (The ECRYPT Stream Cipher Project) [15]. Meanwhile, AKBA refers to new cryptographic algorithms that have not yet been published in recognised standards or widely adopted in the cryptographic community.

The AKSA MySEAL project encompasses various categories of cryptographic primitives, including block cipher, stream cipher, asymmetric cryptographic scheme, cryptographic hash function, prime number generator, and deterministic random bit generator. Block cipher can be further divided into two categories: general-purpose block cipher and lightweight block cipher. Asymmetric cryptographic scheme consists of three classes: asymmetric encryption scheme, digital signature scheme, and key agreement scheme. Whereas, cryptographic hash function can be categorised into general-purpose hash function and lightweight hash function. For prime number generator, there are probabilistic prime generator and deterministic prime generator [24].

The paper is structured as follows. Section 2 introduces the key parties involved in MySEAL evaluation process and discusses the process of MySEAL evaluation for AKSA MySEAL and AKBA MySEAL. In Section 3, the submission and evaluation criteria of AKSA MySEAL and AKBA MySEAL are meticulously described for each primitive. The evaluation result for AKSA MySEAL is presented in Section 4 and Section 6 introduces MySEAL 2.0 as the continuation of the original MySEAL project. Finally, this paper concludes MySEAL initiative in Section 7.

2 MySEAL Evaluation Process

The evaluation process for MySEAL algorithms spanned an eight-month period for AKSA MySEAL and a twenty-four-month period for AKBA MySEAL, during which security and assessment were conducted by onshore and offshore cryptographic experts. In addition to security assessment, the performance of the candidate algorithms was assessed with different measures. The key parties involved in the development of MySEAL include MySEAL Secretariat, which was responsible for providing the supporting documents needed during evaluation process. Meanwhile, the roles of MySEAL Focus group which comprises of national cryptographic experts from Malaysian national agencies and Malaysian universities, were to determine the direction of MySEAL project and to approve MySEAL development activities. For AKBA MySEAL, developers were required to submit novel algorithms that were original and not listed in existing standards or other cryptographic algorithm listing projects. Local cryptographic experts were appointed as AKSA MySEAL Evaluation Panel of Expert and AKBA MySEAL Evaluation Panel of Expert, to conduct thorough assessments on both the existing cryptosystems and the newly proposed cryptosystems, respectively. After the process of evaluating AKSA MySEAL and AKBA MySEAL algorithms ended, the shortlisted algorithms were further analysed by AKSA MySEAL External Reviewer Panel and AKBA MySEAL External Reviewer Panel, respectively. The external reviewer panels were chosen from several international cryptographic specialists.

2.1 AKSA MySEAL Evaluation Process

The timeline of AKSA MySEAL evaluation process can be divided into four intervals: the pre-evaluation phase, the first evaluation phase, the second evaluation phase, and the post-evaluation phase. Firstly, during the pre-evaluation phase which started in 2016, MySEAL Secretariat proposed a selection of cryptographic algorithms listed in recognised standards and cryptographic algorithm listing projects as candidates for AKSA MySEAL. MySEAL Focus Group endorsed highly significant algorithms from these lists for evaluation

in the first phase of AKSA MySEAL evaluation. To evaluate the algorithms, MySEAL Focus Group established the MySEAL Submission and Evaluation Criteria, which serve as benchmarks. MySEAL Secretariat gathered the information for the submission and evaluation criteria by referencing to other established cryptographic algorithm listing projects and reputable organisations such as FIPS, CRYPTREC, NESSIE, and eSTREAM. Then, both submission and evaluation criteria were selected to prioritise and emphasise the security analysis, performance efficiency, flexibility, maturity, and the soundness of justification of a cryptographic algorithm to be evaluated. These criteria were then refined by the MySEAL Focus Group to ensure that they align with the global standards. Amendments were also made to these criteria during evaluation process to ensure they are practical, feasible, and most importantly, suitable for use in assessing cryptographic algorithms.

Section 3 provides more details on the description of MySEAL Submission and Evaluation Criteria. In 2016 and 2018, the MySEAL Project: Submission and Evaluation Criteria Version 1.0 and MySEAL Project: Submission and Evaluation Criteria Version 2.0 were published as the guideline documents for AKSA MySEAL, respectively.

The evaluation process of AKSA MySEAL algorithms commenced with the first phase, which lasted for three months. During this phase, AKSA MySEAL Evaluation Panel of Expert developed the First Phase of AKSA MySEAL Scoring Matrix based on MySEAL Submission Criteria, and evaluated the algorithms endorsed by MySEAL Focus Group during the pre-evaluation phase. In accordance with the scores, a shortlist of AKSA MySEAL candidates was recommended by the experts to MySEAL Focus Group for endorsement. At the end of this phase, the First Phase of AKSA MySEAL Evaluation Report was produced.

The second phase of AKSA MySEAL evaluation comprised thorough assessments and justifications over a period of five months. In the initial stages, AKSA MySEAL Evaluation Panel of Expert formulated the Second Phase of AKSA MySEAL Scoring Matrix, utilizing MySEAL Evaluation Criteria as their guide. Subsequently, a series of evaluations based on the scoring matrix and evaluation criteria, were conducted for the proposed algorithms intended for AKSA MySEAL. Upon completing the assessment, AKSA MySEAL Evaluation Panel of Expert recommended the final list of algorithms that had successfully met the set forth requirements to MySEAL Focus Group. The endorsed final list of algorithms was then submitted to AKSA MySEAL External Reviewer Panel for a comprehensive review and evaluation. The entire process and outcomes of this phase were meticulously documented in the Second Phase of AKSA MySEAL Evaluation Report.

The AKSA MySEAL evaluation process concluded in 2017. In the post-evaluation phase, MySEAL Secretariat published the final list of AKSA MySEAL algorithms. Supporting documents developed for AKSA MySEAL include the Guideline on the Usage of AKSA MySEAL Recommended Cryptographic Algorithms V1 [10] and the Technical Report on the Non-Inclusion Cryptographic Algorithms in AKSA MySEAL (AKSA-NICA) v1.0 [11].

2.2 AKBA MySEAL Evaluation Process

The time interval for AKBA MySEAL evaluation process exceeded that of the AKSA MySEAL evaluation process due to the comprehensive assessment required for newly developed algorithms. The evaluation process of AKBA MySEAL consisted of a pre-evaluation phase, three evaluation phases, and a post-evaluation phase. The first, second, and third evaluation phases each took three, five, and twelve months, respectively. During the pre-evaluation phase, the submission period for AKBA MySEAL algorithms commenced in December 2016 and remained open for twelve months. MySEAL Secretariat recommended a compilation of submitted algorithms to MySEAL Focus Group for approval to be evaluated in the first phase of AKBA MySEAL evaluation.

The first phase of AKBA MySEAL evaluation involved the development of the First

Phase of AKBA MySEAL Scoring Matrix by AKBA MySEAL Evaluation Panel of Experts, based on the MySEAL Submission Criteria. The Panel of Experts then conducted evaluations on the list of submitted algorithms endorsed by MySEAL Focus Group, with reference to the First Phase of AKBA MySEAL Scoring Matrix and MySEAL Submission Criteria. Subsequently, AKBA MySEAL Evaluation Panel of Experts provided the initial list of recommended algorithms for AKBA MySEAL, which required approval from MySEAL Focus Group. At the end of the first phase of AKBA MySEAL evaluation, the First Phase of AKBA MySEAL Evaluation Report was produced.

The shortlisted algorithms underwent further evaluation in the second phase of AKBA MySEAL evaluation. This phase allowed AKBA MySEAL developers to improve and refine their algorithms before MySEAL Secretariat recommended the improved versions to MySEAL Focus Group for evaluation. AKBA MySEAL Evaluation Panel of Expert developed the Second Phase of AKBA MySEAL Scoring Matrix by referring to MySEAL Evaluation Criteria. This matrix was used as a guide for further evaluations of the algorithms. Shortlisted algorithms that were endorsed by MySEAL Focus Group proceeded to the third phase of AKBA MySEAL evaluation. The Second Phase of AKBA MySEAL Evaluation Report was generated.

The Third Phase of AKBA MySEAL Evaluation aimed at further refining the second shortlist of AKBA MySEAL algorithms. This phase was similar to the second phase of AKBA MySEAL evaluation. It involved developing the Third Phase of AKBA MySEAL Scoring Matrix, based on cryptanalysis and security analysis evaluation.The shortlisted algorithms were then evaluated using this scoring matrix, and the final list of AKBA MySEAL algorithms was endorsed by the MySEAL Focus Group before the AKBA MySEAL External Reviewer Panel reviewed the security analysis evaluation. Finally, the Third Phase of AKBA MySEAL Evaluation Report was completed.

The post-evaluation phase focused solely on the publication of final list of AKBA MySEAL by MySEAL Secretariat. The AKBA MySEAL project concluded in 2020.

2.3 Comparison of Evaluation Process

Countries such as the United States, Japan, and various European nations have established their own cryptography standards, namely FIPS, CRYPTREC, and NESSIE. These projects serve as sources of reference when developing the evaluation process for MySEAL.

The evaluation process for FIPS [26] consists of first-phase evaluation, second-phase evaluation, and final evaluation. The main activities during the first and second phases involve public evaluations and intensive discussions of the preliminary candidates' analysis through workshops to produce cut-off lists of algorithms. A standardisation conference is held throughout the first and second phases of evaluations, and a summary report is produced at the end of the second-phase evaluation, containing the possible recommended algorithms for standardisation. The final evaluation focuses on the security levels that can be provided by the algorithms in the presence of practical attacks. Finally, the approved list of algorithms by FIPS is published, and a summary report is produced.

Fundamentally, the CRYPTREC evaluation process [19] involves an initial screening assessment to generate a list of eligible candidates, followed by subsequent in-depth assessments by local and global researchers, and then a public review by research consortia. The screening evaluation encompasses a security evaluation to filter out weak algorithms that are obviously prone to attacks and implementation evaluation, where comprehensive implementation reports on software and hardware by the submitters are required to pass the screening. In-depth assessment allows the shortlisted algorithms to be examined experimentally by experts, and further assessment is opened for public evaluation. After all evaluation activities are concluded, algorithm recommendations are published. To provide assurance on the recommended algorithms, CRYPTREC conducts continuous monitoring.

NESSIE [27] evaluated the submitted cryptographic algorithms by dividing the process into security evaluation and performance evaluation. The initial phase of evaluation, which focuses on the security of algorithms, is an internal process. This phase includes verifying algorithm compliance with the requirements and conducting a comprehensive initial evaluation. When carrying out the algorithms compliance check, it is essential to confirm that there are no obvious weaknesses in the algorithms. The initial evaluation is divided into two independent assessments, and at the end of this phase, a unified summary of the assessments is produced. Only algorithms that pass both the compliance check and the independent assessments will be further evaluated in the second phase by external evaluators through an open workshop. The main focus of the open workshop is to carry out security and performance evaluations on the narrowed-down candidates. Upon completion of the second phase, a comprehensive report covering the security and performance of eligible algorithms is published. The report provides an overview of common attacks, security assurances, performance assurances, and a conceptual analysis of the performance of the selected algorithms.

The evaluation process of MySEAL is not vastly different from those used in FIPS, CRYPTREC, and NESSIE. Specifically, there are two detailed evaluations for AKSA MySEAL and three detailed evaluations for AKBA MySEAL, focusing on the security and performance of algorithms to their credibility. Algorithms for AKSA MySEAL and AKBA MySEAL undergo a basic compliance check or initial screening during the first-phase evaluation. A summary report is produced at the end of each evaluation phase to describe the evaluation process and outcomes. Unlike the FIPS, CRYPTREC, and NESSIE, MySEAL does not include a public review. However, external evaluation by notable foreign researchers strengthens the validity of algorithms approved by MySEAL. To conclude the whole process, MySEAL publishes a guideline and a technical report on the selected algorithms for AKSA MySEAL and AKBA MySEAL.

The evaluation processes of FIPS, CRYPTREC, NESSIE, and MySEAL are summarised in Table 1.

3 MySEAL Criteria

This section describes the submission and evaluation criteria for each primitive during the evaluation processes of AKSA MySEAL and AKBA MySEAL.

3.1 Submission Criteria

During the first phase of MySEAL evaluation, all cryptographic algorithms were assessed against a set of submission criteria. These criteria primarily focused on the structural requirements of the algorithms, basic security analysis, the presence of test vectors, and the designer's justification of design principles. Additionally, reports on implementation and performance for the targeted software (program code size and RAM size) and/or targeted hardware (chip area, cycle, bits per cycle, power, and energy) were mandated. It is worth noting that there are some special criteria which apply to specific primitive groups.

3.1.1 Symmetric Block Cipher

Block ciphers considered for MySEAL are categorised into general-purpose and lightweight applications. Firstly, general-purpose block cipher primitives must have a minimum key length and block length of 128 bits, while for lightweight block ciphers must have at least 80 bits for the key length and 64 bits for the block length. Secondly, the security analysis report should include, but not be limited to, FIPS statistical tests, linear cryptanalysis, and differential cryptanalysis. Designers of the cipher should provide a minimum of three test

Table 1: Summary of the Evaluation Process for Various Cryptography Projects

Standard	Evaluation Process
FIPS	1. First-phase evaluation: - public evaluation - first workshop - first standardisation conference 2. Second-phase evaluation: - public evaluation - second workshop - second standardisation conference - produce a summary report and a list of shortlisted algorithms 3. Final evaluation: - produce a summary report - publication of approved list of algorithms
CRYPTREC	1. Initial screening assessment - security and performance assessments 2. In-depth assessments by local and global researchers 3. Public review by research consortia 4. Publication of approved algorithm recommendations 5. Continuous monitoring
NESSIE	1. First-phase evaluation (Internal process): - algorithm compliance check - comprehensive initial evaluation: • two independent assessments • a unified summary of both assessments - produce a list of shortlisted algorithms 2. Second-phase evaluation (External process): - shortlisted algorithms go through an open workshop - publication of security and performance evaluation report
MySEAL	1. Pre-evaluation phase: - algorithms selection to be evaluated 2. First-phase evaluation - algorithm compliance check - produce a summary report and a list of shortlisted algorithms 3. Second-phase evaluation: - detailed evaluation on security and performance of algorithms - external evaluation by notable foreign researchers (only for AKSA MySEAL) - produce a summary report and a list of shortlisted algorithms 4. Third-phase evaluation (only for AKBA MySEAL): - detailed evaluation on security and performance of algorithms - external evaluation by notable foreign researchers - produce a summary report and a list of shortlisted algorithms 5. Post-evaluation phase: - publication of approved list of algorithms - produce a guideline and a technical report

vectors for each key size and three plaintext-ciphertext pairs for each key. The processing sample must be in ECB mode with '0' bit padding, and intermediate output should be provided for each round.

3.1.2 Symmetric Stream Cipher

The first requirement for a stream cipher is that it must operate either in synchronous or self-synchronous mode. When implementing the cipher in hardware, the minimum key length and internal memory should be 80 bits and 160 bits, respectively. In software implementations, these values should be at least 128 bits for the key length and 256 bits for the internal memory. Additionally, a comprehensive security analysis report is mandatory. This report should encompass various aspects, including FIPS statistical tests, algebraic attacks, correlation attacks, distinguishing attacks, and guess-and-determine attacks. Finally, test vectors including three sets of keys for each key size, three Initialisation Vectors (IVs) for each key, a keystream length of 256 bits, and the internal state after generating 256 keystream bits are needed.

3.1.3 Asymmetric Cryptographic Primitive

For asymmetric cryptographic primitive, proof of correctness is of paramount importance. The security analysis should include, but is not limited to hard mathematical problems and assumptions. A minimum key length of 2^{128} is necessary to attain the security level, as well as the security model and its accompanying proof. Finally, the inclusion of comprehensive test vectors is mandatory. The test vectors should comprise a minimum of three key pairs and two processing samples for each key pair.

3.1.4 Cryptographic Hash Function Primitive

Two classes in cryptographic hash function primitive are general-purpose hash function and lightweight cryptographic hash function. For the former, the digest sizes must include 224 bits, 256 bits, 384 bits, 512 bits, or larger. While for the latter, the digest sizes must be 80 bits, 128 bits, and 160 bits. Both classes share a maximum message length of $2^{64} - 1$ bits. Additionally, the security analysis report should include, but is not limited to, pre-image resistance, second pre-image resistance and collision resistance. Moreover, the primitive's designers should provide test vectors, including a minimum of three samples for each data size and the intermediate state for each round.

3.1.5 Prime Number Generator Primitive

The prime number generator (PNG) primitive includes two types of generator: probabilistic prime generator and deterministic prime generator. Both categories should have polynomial running times.

Probabilistic prime generator should achieve a proven correctness rate of at least 75% (comparable to the Miller-Rabin Primality test). The primality test should yield one of the following outcomes: 'input is prime', 'input is composite', or 'test inconclusive'. Test vectors are required, with prime sizes of 512, 1024, 2048, 3072, 4096, 7680, and 15360 bits, and three seeds: 128, 256 and 512 bits, for each prime size.

For deterministic prime generator, proof of correctness, the capability to distinguish Carmichael numbers from prime numbers, and the ability to generate pseudo-prime samples from the generator are necessary. Additionally, FIPS statistical tests are also required.

3.1.6 Deterministic Random Bit Generator Primitive

For the deterministic random bit generator (DRBG) primitive, there are DRBGs based on either symmetric or asymmetric methodologies, or neither of them. All three types of DRBG must run in polynomial time. Additionally, Furthermore, FIPS statistical tests are also required and proof of correctness for DRBGs that are not based on symmetric methodologies is mandotory.

If the primitive's internal state of DRBGs based on asymmetric methodologies contains n bits, its period should be at least 2^n. The seed size of test vector should be 128, 256, and 512 bits, utilizing strong asymmetric parameters. For example, the prime numbers used in the Integer Factorisation Problem (IFP) should be of 1024, 2048, 4096 bits, while the prime used in the Discrete Logarithm Problem (DLP) should also be of 1024, 2048, 4096 bits.

For DRBGs based on symmetric methodologies, test vectors with seed sizes of 128, 256, and 512 bits, utilizing strong symmetric constructions such as AES and TDES, are required.

For DRBGs that are not based on both methodologies, designers should provide a justification for the algorthm's design principles, and the test vectors should include seed sizes of 128, 256, and 512 bits.

3.2 Evaluation Criteria

This section outlines the evaluation criteria for each primitive, primarily categorised into security analysis, cost and performance analysis, and implementation characteristics. Design simplicity, characterised by an elegant, concise, neat, and easily understandable design, is considered as an added advantage. Additionally, each primitive must provide a sound justification for the algorithm's design principles. Certain primitive groups might also have minor criteria specific to their category.

3.2.1 Block Cipher

A block cipher should achieve a security level that matches its key size based on cryptanalysis attacks. For example, a 128-bit key should provide a 128-bit security level. Additionally, the cipher must pass all FIPS statistical tests across nine data categories.

For cost and performance analysis, AES cipher is used as a benchmark to evaluate computational efficiency and memory requirements of block ciphers. For lightweight block ciphers, hardware and/or software implementation measurements should be comparable to the PRESENT cipher.

In terms of implementation characteristics, the flexibility of the algorithm may include, but is not limited to, the ability to accommodate additional key and block sizes. Block cipher algorithms should be implemented securely and efficiently across a wide variety of platforms and applications, as well as be able to operate as stream cipher, pseudorandom number generator (PRNG), message authentication code (MAC) generator or hash function. Furthermore, suitability of the cipher in both software and hardware is considered as an added advantage, as it demonstrates implementation efficiency in various platforms.

3.2.2 Stream Cipher

The criteria for security analysis and implementation characteristics for stream cipher are similar to those for block cipher. ChaCha20 algorithm serves as a benchmark for evaluating the computational efficiency and memory requirements of stream cipher. It is important to note that, unlike block cipher, stream cipher is specifically expected to operate as PRNG only.

3.2.3 Asymmetric Cryptographic Primitive

For asymmetric cryptographic primitive, the security criteria include various aspects. The hard mathematical problems of an asymmetric scheme must be grounded on either conventional problems, such as DLP and its variants, or post-quantum problems, such as lattice-based problems and code-based problems. The minimum security level that a

scheme needs to achieve is 2^{128}. Moreover, the correctness of the security model and its proof must be in line with the assumptions made in the hard problems.

In terms of cost and performance analysis, parameter size for each security level such as key size and ciphertext size for encryption, key size, number of passes and bandwidth for key agreement, as well as key size and signature size for digital signature, are important. Computational complexity is also an imperative aspect that is meticulously considered during the evaluation process.

Furthermore, asymmetric cryptographic schemes should provide a valid proof of correctness, and comparison analysis is considered advantageous.

3.2.4 Cryptographic Hash Function Primitive

Hash function primitive should achieve a security level comparable to its digest size from a cryptanalytic attack perspective, such as a 128-bit security level for a 128-bit digest size.

For cost and performance analysis, SHA-3 hash function is used as a benchmark to access the computational efficiency and memory requirements, while for lightweight hash functions, hardware and/or software implementation measurements should be comparable to SPONGENT.

Concerning the implementation characteristics, the flexibility of hash function may include but not limited to: the ability to accommodate additional digest sizes and the ability to be implemented securely and efficiently in a wide variety of platforms and applications.

3.2.5 Prime Number Generator Primitive

The algorithms for prime number generator (PNG) primitive should meet the following security requirements: safe pseudo-random prime number generator, produce strong prime number and strong prime number pair for integer factorisation problem (IFP safe) and produce strong prime number for discrete logarithm problem (DLP safe). Additionally, the algorithms must successfully pass all FIPS statistical tests and demonstrate polynomial time running time. In terms of implementation characteristics, the algorithm's flexibility may include, but is not limited to, the ability to accommodate additional extendable block sizes, generate primes within a specified interval, and can be implemented efficiently across a wide range of platforms and applications. Furthermore, merit criteria include algorithm suitability for both software and hardware, as well as comparative analysis with existing prime number generators.

3.2.6 Deterministic Random Bit Generator Primitive

The algorithms for deterministic random bit generator (DRBG) primitive should adhere to the following security requirements: they should ensure safety as a pseudo-random number generator. Particularly, the algorithms are safe in asymmetric cryptographic environments (IFP and DLP safe) and symmetric cryptographic environments. Additionally, the algorithms must successfully pass all FIPS statistical tests and undergo security analysis with respect to seed entropy. Next, the algorithms should demonstrate polynomial time running time. Overall, the evaluation criteria for DRBG shares similar requirements with PNG primitive, but comparative analysis should be conducted with existing random number generators.

3.3 Comparison of Evaluation Criteria

Evaluation criteria are important when evaluating algorithms. They determine the integrity and reliability of algorithms approved by standards. Both the quality and quantity of the adopted evaluation criteria play significant roles in ruling out algorithms with lower

integrity and reliability.

Security is the most critical criterion for evaluating algorithms; hence, all FIPS, CRYPTREC, NESSIE, and MySEAL projects take this criterion into account. FIPS and CRYPTREC focus on integrated evaluation of algorithms against common attacks, systematic attacks, and heuristic attacks. NESSIE classifies the security level of each cryptographic primitive as normal, high, or normal-legacy, depending on the minimum key length required and other security criteria such as block length and output length.

Moreover, implementation characteristics are also adopted as evaluation criteria by FIPS, CRYPTREC, NESSIE, and MySEAL to assess the efficiency and suitability of algorithms in both software and hardware, ensuring optimal performance and compatibility across different platforms.

Furthermore, FIPS, CRYPTREC, NESSIE, and MySEAL include performance as part of their evaluation criteria, with submissions assessed based on various performance indicators as appropriate. However, only MySEAL and FIPS include cost as an evaluation criterion. Latency, throughput, and power consumption are considered performance indicators, while area, memory, and energy consumption are considered cost indicators.

NESSIE and MySEAL added simplicity and clarity of design as its evaluation criteria, which are not included in the other standards. Simplicity and clarity in algorithm design contribute to easier implementation, maintenance, and verification, as well as a lower likelihood of introducing errors or vulnerabilities.

MySEAL evaluates criteria more critically by considering the soundness of justification for the algorithm's design principles, the validity proof of correctness, and the advantages of comparison analysis.

The evaluation criteria of FIPS, CRYPTREC, NESSIE, and MySEAL are summarised in Table 2.

Table 2: Standards comparison criteria.

Standard	**FIPS**	**CRYPTREC**	**NESSIE**	**MySEAL**
Security	✓	✓	✓	✓
Cost	✓			✓
Performance	✓	✓	✓	✓
Implementation characteristics	✓	✓	✓	✓
Simplicity and clarity of design			✓	✓
Soundness of justification on algorithm's design principles				✓
Valid proof of correctness				✓
Comparison analysis is an advantage				✓

4 AKSA MySEAL Evaluation Result

A total of 48 algorithm candidates were considered during the first phase of evaluation. 42 out of 48 algorithms were then recommended to be further evaluated in the second phase of evaluation. The final list of AKSA MySEAL which successfully passed both phases consists of 30 cryptographic algorithms, as shown in Table 3.

Table 3: Listed algorithms in AKSA MySEAL.

Cryptographic Primitive	Algorithm	
Symmetric Block Cipher	Block Cipher: 1. AES-128, AES-192, AES-256 [30] 2. Camellia-128, Camellia-192, Camellia-256 [1] 3. CLEFIA-128, CLEFIA-192, CLEFIA-256 [33] Lightweight Block Cipher: 1. PRESENT-80, PRESENT-128 [8] 2. HIGHT [18]	
Symmetric Stream Cipher	1. ChaCha20-256 [5] 2. KCipher-2 [21] 3. Rabbit [6]	
Asymmetric Cryptographic Scheme	Digital Signature Scheme: 1. DSA [28] 2. ECDSA [20] 3. RSA-PSS [4] Key Agreement Scheme: 1. ECDH [9] 2. DH [13]	Asymmetric Encryption Scheme: 1. PSEC-KEM [32] 2. RSA-KEM [34] 3. ACE-KEM [34] 4. ECIES-KEM [34] 5. RSA-OAEP [3] 6. NTRU [17]
Cryptographic Hash Function	Hash Function: 1. SHA-384, SHA-512, SHA-512/224, SHA-512/256 [12] 2. SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, SHAKE256 [14] Lightweight Hash Function: 1. SPONGENT-88, SPONGENT-128, SPONGENT-160, SPONGENT-224, SPONGENT-256 [7] 2. PHOTON-80/20/16, PHOTON-128/16/16, PHOTON-160/36/36, PHOTON-224/32/32, PHOTON-256/32/32 [16]	
Prime Number Generator	1. Miller-Rabin Primality Test [29] 2. Elliptic curve Primality Certificate [2] 3. Shawe-Taylor's Algorithm [31]	
Deterministic Random Bit Generator	1. HMAC-SHA-384-DRBG, HMAC-SHA-512-DRBG [25] 2. SHA-512/224-DRBG, SHA-512/256-DRBG5, SHA-384-DRBG, SHA-512-DRBG [25] 3. AES-128-CTR-DRBG, AES-192-CTR-DRBG, 3-Key-TDEA-CTR-DRBG [25]	

5 AKBA MySEAL Evaluation

Seven newly developed cryptographic algorithms were recommended for AKBA MySEAL in the pre-evaluation phase. These comprise one block cipher algorithm, three algorithms from asymmetric encryption schemes, two algorithms from digital signature schemes, and one prime number generator algorithm. The algorithms underwent at most three evaluation phases as described in Section 2.2.

5.1 AKBA MySEAL Evaluation Result

All seven cryptographic algorithm candidates for AKBA MySEAL successfully passed the First Phase of AKBA MySEAL Evaluation. However, during the Second Phase of AKBA MySEAL Evaluation, only one algorithm from the Asymmetric Encryption Scheme primitive was eliminated.

Out of the seven algorithms submitted for AKBA MySEAL, only two passed through to the Third Phase of AKBA MySEAL Evaluation, which are from asymmetric signature scheme and digital signature scheme, respectively.

6 MySEAL 2.0

MySEAL initiative began in 2016 and concluded in 2020 with a successfully publication of the AKSA MySEAL in 2017. In 2023, after more than five years later, a review of the AKSA MySEAL list was carried out, marking the start of a continual initiative named MySEAL 2.0. MySEAL 2.0 encompasses of three distinct categories; AKSA MySEAL Approved, AKSA MySEAL Neutral, and AKSA MySEAL Monitored. AKSA MySEAL Approved lists existing cryptographic algorithms selected from various standards (i.e. FIPS and ISO/IEC) and other cryptographic algorithm listing projects. AKSA MySEAL Neutral lists cryptographic algorithms that are in a transitional phase, allowing their usage with caution under controlled circumstances. However, these algorithms are listed in other approved cryptographic algorithm standardisation projects. AKSA MySEAL Monitored are list of cryptographic algorithms intended solely for ensuring interoperability with legacy systems.

The processes and activities for AKSA MySEAL 2.0 Approved initiative were carried out similarly to those activities caried out during the first round of AKSA MySEAL initiative. It commenced with the pre-evaluation process, which included refining the Nomination Criteria (previously referred to as Submission Criteria) and the Evaluation Criteria, as well as selecting the most current cryptographic algorithm listed in recognised standards and cryptographic algorithm listing project.

The first phase of AKSA MySEAL 2.0 Approved evaluation was conducted in May 2023, whereas the second phase was completed in December 2023. Scoring matrixes were developed by the MySEAL Evaluation Panel of Expert based on the MySEAL Nomination Criteria and MySEAL Evaluation Criteria for first phase and second phase respectively. Evaluation reports were produced from the two evaluation phases. To ensure the report prepared by cryptographic experts throughout Malaysia meets the global standards, the Second Phase of AKSA MySEAL Approved 2.0 Evaluation Report was submitted to MySEAL External Reviewers from Germany, New Zealand, France, Belgium, and Iraq for thorough review.

To move forward, new cryptographic algorithms, AKBA MySEAL 2.0 need to be incorporated into the MySEAL 2.0 initiative. Algorithms submitted to the AKBA MySEAL 2.0 initiative will undergo three evaluation phases. The algorithms for AKBA MySEAL 2.0 are sourced through public submissions. An announcement to the public cryptographic community to call for algorithm submissions for AKBA MySEAL 2.0 is necessary and has

been made in March 2024. Additionally, for the AKSA MySEAL initiatives, it has been proposed incorporating new cryptographic primitives to support the extensive implementation of cryptographic algorithms in cryptographic products. The new cryptographic primitives to be included in the AKSA MySEAL 2.1 are Message Authentication Code (MAC), Key Derivation Function (KDF), Threshold Cryptography, Homomorphic Encryption, and Authenticated encryption.

7 Conclusion

The MySEAL projects, encompassing both the AKSA and AKBA initiatives, is focused on establishing a robust portfolio of cryptographic algorithms for Malaysia. Both initiatives involved multi-stage evaluations conducted by a panel of cryptographic experts, culminating in the finalisation of the AKSA MySEAL list in 2017 and the AKBA MySEAL list in 2020.

Furthermore, MySEAL 2.0 was launched to ensure the continued relevance of the AKSA MySEAL-approved algorithms. The evaluation process mirrors the original MySEAL project but introduces three categories (AKSA MySEAL Approved, AKSA MySEAL Neutral, and AKSA MySEAL Monitored) to classify the evaluated algorithms. Looking ahead, newly developed cryptographic algorithms will be incorporated into AKBA MySEAL 2.0, as announced in March 2024, and new cryptographic primitives will be added to AKSA MySEAL 2.1 to support secure cryptographic products

Overall, the MySEAL projects align with the National Cryptography Policy, reinforcing Malaysia's cryptographic autonomy and security infrastructure. By leveraging on established cryptographic standards and fostering new algorithms, MySEAL enhances the nation's information security systems and cybersecurity posture. The projects carefully evaluate cryptographic primitives based on criteria such as security, performance, and implementation characteristics, playing a pivotal role in safeguarding sensitive data in an interconnected world.

References

[1] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms—design and analysis. In *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000 Waterloo, Ontario, Canada, August 14–15, 2000 Proceedings 7*, pages 39–56. Springer, 2001.

[2] A Oliver L Atkin and François Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61(203):29–68, 1993.

[3] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings 13*, pages 92–111. Springer, 1995.

[4] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *International conference on the theory and applications of cryptographic techniques*, pages 399–416. Springer, 1996.

[5] Daniel J Bernstein et al. Chacha, a variant of salsa20. In *Workshop record of SASC*, volume 8, pages 3–5. Citeseer, 2008.

[6] Martin Boesgaard, Mette Vesterager, Thomas Pedersen, Jesper Christiansen, and Ove Scavenius. Rabbit: A new high-performance stream cipher. In *Fast Software*

Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10, pages 307–329. Springer, 2003.

[7] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13*, pages 312–325. Springer, 2011.

[8] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*, pages 450–466. Springer, 2007.

[9] D Brown. Standards for efficient cryptography, sec 1: elliptic curve cryptography. *Released Standard Version*, 1, 2009.

[10] CyberSecurity Malaysia. Guideline on the usage of aksa myseal recommended cryptographic algorithms. https://www.cybersecurity.my/data/content_files/56/2085.pdf, 2020. Accessed: 2024-07-29.

[11] CyberSecurity Malaysia. Technical report on the non-inclusion cryptographic algorithms in aksa myseal (aksa-nica) v1.0. https://www.cybersecurity.my/data/content_files/56/2569.pdf, 2020. Accessed: 2024-07-29.

[12] Quynh H Dang. Secure hash standard. *NIST Pubs*, 2015.

[13] Whitfield Diffie and Martin E Hellman. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, pages 365–390. IEEE, 2022.

[14] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. *NIST Pubs*, 2015.

[15] eSTREAM Project. estream: The ecrypt stream cipher project. https://www.ecrypt.eu.org/stream/, 2024. Accessed: 2024-07-29.

[16] Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 222–239. Springer, 2011.

[17] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ntru: A ring-based public key cryptosystem. In *International algorithmic number theory symposium*, pages 267–288. Springer, 1998.

[18] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, et al. Hight: A new block cipher suitable for low-resource device. In *Cryptographic Hardware and Embedded Systems-CHES 2006: 8th International Workshop, Yokohama, Japan, October 10-13, 2006. Proceedings 8*, pages 46–59. Springer, 2006.

[19] H. Imai. Evaluation of cryptgraphic techniques in japan, cryptrec project, 2001.

[20] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1:36–63, 2001.

[21] Shinsaku Kiyomoto, Toshiaki Tanaka, and Kouichi Sakurai. K2 stream cipher. In *E-business and Telecommunications: 4th International Conference, ICETE 2007, Barcelona, Spain, July 28-31, 2007, Revised Selected Papers 4*, pages 214–226. Springer, 2009.

[22] CyberSecurity Malaysia. National trusted cryptographic algorithm list (myseal). https://mykripto.cybersecurity.my/index.php/services/myseal, 2024. Accessed: 2024-07-29.

[23] MyKripto Cybersecurity. Akba myseal. https://mykripto.cybersecurity.my/index.php/services/myseal/myseal-category/akba-myseal, 2024. Accessed: 2024-07-29.

[24] MyKripto Cybersecurity. Aksa myseal. https://mykripto.cybersecurity.my/index.php/services/myseal/myseal-category/aksa-myseal, 2024. Accessed: 2024-07-29.

[25] National Institute of Standards and Technology (NIST). Recommendation for random number generation using deterministic random bit generators. Technical Report NIST SP 800-90Ar1, National Institute of Standards and Technology, 2015. Accessed: 2024-07-29.

[26] National Institute of Standards and Technology (NIST). Federal information processing standards (fips). https://csrc.nist.gov/publications/fips, 2024. Accessed: 2024-07-29.

[27] Bart Preneel. New european schemes for signature, integrity and encryption (nessie): a status report. In *International Workshop on Public Key Cryptography*, pages 297–309. Springer, 2002.

[28] FIPS PUB. Digital signature standard (dss). *FIPS PUB*, pages 186–192, 2000.

[29] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

[30] Vincent Rijmen and Joan Daemen. Advanced encryption standard. *Proceedings of federal information processing standards publications, national institute of standards and technology*, 19:22, 2001.

[31] J Shawe-Taylor. Generating strong primes. *Electronics Letters*, 16(22):875–877, 1986.

[32] Rachel Shipsey. Psec-kem. *Public report, NESSIE*, page 199, 2001.

[33] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher clefia. In *Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14*, pages 181–195. Springer, 2007.

[34] Victor Shoup. A proposal for an iso standard for public key encryption. *Cryptology ePrint Archive*, 2001.

Institute for Mathematical Research
Universiti Putra Malaysia,
43400 UPM, Serdang, Selangor,
MALAYSIA.
https://inspem.upm.edu.my