AIMS Mathematics, 10(4): 8569–8586.
DOI: 10.3934/math.2025393
Received: 30 December 2024
Revised: 13 March 2025
Accepted: 07 April 2025
Published: 14 April 2025

https://www.aimspress.com/journal/Math

Research article

A fast semiring-based public-key encryption

Dieaa I. Nassr[1], Hatem M. Bahig[2], Mohamed A. G. Hazber[3], Ibrahim M. Alseadoon[3] and Hazem M. Bahig[3,*]

[1] Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt

[2] College of Computer and Information Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia

[3] Department of Information and Computer Science, College of Computer Science and Engineering, University of Ha'il, Ha'il 81481, Saudi Arabia

* **Correspondence:** Email: h.bahig@uoh.edu.sa.

Abstract: This paper presents a new public-key encryption with security based on the two-sided digital circulant matrix action problem defined over the semiring proposed by Huang et al. [14]. The performance evaluation of the proposed encryption scheme shows its robustness and efficiency compared to state-of-art encryption schema. We also provide a security analysis of the proposed encryption. It is suitable for post-quantum cryptography and IoT platforms.

Keywords: circulant matrix; public-key encryption; post-quantum; semiring
Mathematics Subject Classification: 15A80, 94A60

1. Introduction

Many public-key cryptosystems and protocols have been proposed since Diffie and Hellman [9] introduced the idea of public-key cryptography in 1976. Many applications use public-key cryptosystems and protocols, such as SSH [28], OpenPGP [4], and SSL/TLS [8]. The security of public-key cryptosystems and protocols depends on a trapdoor one-way function and the algebraic structure defined on it. The algebraic structures can be commutative or non-commutative. For examples:

- The security of RSA [23] and its variants [16] depends on the difficulty of the integer factorization problem (IFP). The algebraic structure is commutative, such as a ring Z_n, where $n = pq$ is the product of two large primes p, q; or a ring $Z_n[x]$ [5]. Shor [24] showed that quantum computers solve IFP in polynomial time.

- The security of Diffie-Hellman [9], ElGaml [11] and their variants [12] is based on the discrete logarithm problem (DLP) in a cyclic group, in particular, an elliptic curve [12] over a finite field F_q. It is expected that the DLP can be solved in polynomial time using quantum computers [24].
- The security of NTRU [13] lies in the difficulty of hard problems in lattice reduction called the shortest vector and the closest vector problems [6, 13, 19, 20]. The NTRU is defined on a ring of polynomials $Z_q[X]/(X^n - 1)$. This class of cryptosystems and protocols is resistant to quantum attacks.
- The security of some key exchange protocols, such as Anshel et al. [3] and Ko et al. [17], is based basically on the hardness of the conjugator search problem (CSP) defined on non-commutative groups G: given two elements, $g_1, g_2 \in G$, find an element $h \in G$, such that $g_1 = hg_2h^{-1}$. Shpilrain-Ushakov [25] noted that instead of using CSP, we can use the double coset problem (DCP): given $g_1, g_2 \in G$, find $h_1, h_2 \in G$ such that $g_1 = h_1g_2h_2$.

One of the algebraic structures that is recently used in public-key cryptography is semiring [10]. In [7], Grigoriev and Shpilrain suggested using a tropical semiring in public-key cryptosystems. In particular, they showed that solving systems of polynomial equations in a tropical semiring is NP-hard. Huang et al. [15] suggested a public-key encryption and key exchange protocol based on tropical circular matrices. Their security is based on the difficulty of solving tropical non-linear systems of integers. Durcheva [18] proposed a robustness and efficiency key exchange protocol suitable for IoT platforms with security based on the matrix decomposition problem.

Alhussaini and Sergeev [1] used the max-min semiring and the max-T semiring to improve the tropical Stickel key exchange protocol [26]. H. Huang et al. [14] presented a new matrix semiring and extended the Diffie-Hellman key exchange protocol [9] using the new matrix semiring. The extended protocol is suitable for post-quantum cryptography. Its security is based on solving two problems: Problems 2.4 and 2.5, which can be solved by solving quadratic polynomial systems over the presented matrix semiring, which is an NP-hard problem.

In this paper, we are interested in designing a public-key encryption using the semiring defined in [14]. The security of the presented encryption relies on two hard problems: Problems 2.4 and 2.5, see Section 2.

This paper is organized as follows: In Section 2, we review the semiring with a matrix structure and some related hard problems. In Section 3, we present the proposed public-key encryption and prove its correctness. An example of encrypting a message is presented in Section 4. The security analysis of the proposed encryption is presented in Section 5. Section 6 shows the performance of the proposed public-key encryption. Section 7 includes the conclusion.

2. Preliminaries

This section provides a necessary background of a semiring with a matrix structure [14, 27] and computational problems based on the semiring.

Definition 2.1. *Let G be a nonempty set, $\boxplus$ and $\boxtimes$ be two binary operations (addition and multiplication) defined on G. An algebraic structure $(G, \boxplus, \boxtimes)$ is called a semiring if:*

1) $(G, \boxplus)$ is a monoid and commutative with an identity element denoted by 0_G;
2) $(G, \boxtimes)$ is a monoid with an identity element denoted by 1_G;

3) Multiplication distributes over addition from both sides:

$$g_1 \boxtimes (g_2 \boxplus g_3) = (g_1 \boxtimes g_2) \boxplus (g_1 \boxtimes g_3),$$
$$(g_1 \boxplus g_2) \boxtimes g_3 = (g_1 \boxtimes g_3) \boxplus (g_2 \boxtimes g_3);$$

4) $0_G \boxtimes g = g \boxtimes 0_G = 0_G$ *for all* $g \in G$;
5) $0_G \neq 1_G$.

It is not necessary in a semiring that the additive inverse of an element $g \in G$ exists. In the case where an additive inverse in G exists, then G is called a ring. If $\boxtimes$ is commutative, then the semiring $(G, \boxplus, \boxtimes)$ is commutative.

Let N be the set of natural numbers with zero and $G = N \cup \{\infty\}$. Then $(G, \boxplus, \boxtimes)$ is a semiring [14], where the binary operations $\boxplus$ and $\boxtimes$ over $G = N \cup \infty$ are defined as follows:

$$g_1 \boxplus g_2 = \begin{cases} g_1 & \text{if } \delta(g_2) < \delta(g_1), \\ g_2 & \text{if } \delta(g_2) > \delta(g_1), \\ \max(g_1, g_2) & \text{if } \delta(g_1) = \delta(g_2), \end{cases}$$
$$g_1 \boxtimes g_2 = \begin{cases} g_1 & \text{if } \delta(g_1) < \delta(g_2), \\ g_2 & \text{if } \delta(g_1) > \delta(g_2), \\ \min(g_1, g_2) & \text{if } \delta(g_1) = \delta(g_2), \end{cases}$$

where $\delta(g)$ is the sum of all digits of g if $g \in N$; otherwise, i.e., $g = \infty$, $\delta(g) = \infty$.

For example, $\delta(92834) = 9 + 2 + 8 + 3 + 4 = 26$.

Definition 2.2. *[14] Let* $(G, \boxplus, \boxtimes)$ *be a semiring and* $M_n(G)$ *be the set of all* $n \times n$ *matrices over* G. *If* $A = (a_{ij}), B = (b_{ij}) \in M_n(G)$, *then*

1) $A \boxplus B = (a_{ij} \boxplus b_{ij})$.
2) $A \boxtimes B = \left((a_{i0} \boxtimes b_{0j}) \boxplus (a_{i1} \boxtimes b_{1j}) \boxplus \ldots \boxplus (a_{i(n-1)} \boxtimes b_{(n-1)j})\right)$.

The identity elements for addition, 0_M, and multiplication, 1_M, are

$$0_G = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

$$1_G = \begin{bmatrix} \infty & 0 & \cdots & 0 \\ 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \infty \end{bmatrix}.$$

Definition 2.3. *A matrix $A \in M_n(G)$ is called a circulant if A is of the following form:*

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-2} \\ a_{n-2} & a_{n-1} & \cdots & a_{n-3} \\ \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \cdots & a_0 \end{bmatrix}.$$

The circulant matrix A can be represented by the first row of A as follows: $A = [a_0, a_1, \cdots, a_{n-1}]$. Each row of the remaining rows of A is a right cyclic shift (right rotation) of the previous row.

Let $A, B \in M_n(G)$ be two circulant matrices. Then $A \boxtimes B = B \boxtimes A$ is circulant [14]. For example, let

$$A = \begin{bmatrix} 274 & 512 & 571 & 241 \\ 241 & 274 & 512 & 571 \\ 571 & 241 & 274 & 512 \\ 512 & 571 & 241 & 274 \end{bmatrix}, B = \begin{bmatrix} 361 & 350 & 274 & 536 \\ 536 & 361 & 350 & 274 \\ 274 & 536 & 361 & 350 \\ 350 & 274 & 536 & 361 \end{bmatrix}.$$

Then

$$A \boxtimes B = B \boxtimes A = \begin{bmatrix} 274 & 571 & 274 & 274 \\ 274 & 274 & 571 & 274 \\ 274 & 274 & 274 & 571 \\ 571 & 274 & 274 & 274 \end{bmatrix}.$$

The semiring given in Definition 2.2 is used to propose the key exchange protocol [14] with security based on the following two hard problems.

Problem 2.4 (MAP [14])**.** *Let $P_1, P_2 \in M_n(G)$ be two circuant matrices, and $U = P_1 \boxtimes T \boxtimes P_2$ for an arbitrary matrix $T \in M_n(G)$. By giving U and T, the challenge is to obtain two circulant matrices P_1 and P_2 such that $U = P_1 \boxtimes T \boxtimes P_2$.*

Problem 2.5 (CMAP [14])**.** *Let $P_1, P_2, B_1, B_2 \in M_n(G)$ be circuant matrices, $U = P_1 \boxtimes T \boxtimes P_2$, and $V = B_1 \boxtimes T \boxtimes B_2$ for an arbitrary matrix $T \in M_n(G)$. By giving U, V, and T, the challenge is to obtain $K = P_1 \boxtimes B_1 \boxtimes T \boxtimes B_2 \boxtimes P_2$.*

Problem 2.6 (Decision MAP)**.** *Given $T, U \in M_n(G)$. Are there two circulant matrices P_1 and P_2 such that $U = P_1 \boxtimes T \boxtimes P_2$.*

In [14], it is shown that the solution of CMAP is identical to obtaining the matrix K from the known parameters of the protocol. In addition, MAP can be transformed into the problem of solving quadratic polynomial systems on the semiring $(G, \boxplus, \boxtimes)$.

Proposition 2.7. *[14] MAP can be transformed to the problem of solving quadratic polynomial systems on the semiring $(G, \boxplus, \boxtimes)$.*

Therefore, solving such problems is NP-hard.

3. The proposed public-key encryption

In this section, we propose a public-key encryption. The algebraic structure for the proposed encryption is $M_n(G)$. This section consists of two subsections. In Section 3.1, we describe the key generation for the proposed encryption. In Section 3.2, we present the encryption and decryption processes. We also provide the correctness of the proposed encryption.

3.1. Key generation

The private-key consists of two circulant matrices $P_1, P_2 \in M_n(G)$, while the public-key consists of two matrices $S, T \in M_n(G)$, where $T = P_1 \boxtimes S \boxtimes P_2$.

Algorithm 1 describes the generation of public and private keys for the proposed encryption.

Algorithm 1. ***Key generation***
Input:

– *n: matrix dimension*

Output: *Public matrices $S, T \in M_n(G)$ and private circulant matrices $P_1, P_2 \in M_n(G)$.*
Begin

1: *Generate randomly a matrix $S \in M_n(G)$.*
2: *Generate randomly two circulant matrices $P_1, P_2 \in M_n(G)$.*
3: *Compute $T = P_1 \boxtimes S \boxtimes P_2$.*
4: ***Return*** *(S, T), and (P_1, P_2) as the public-key and private-key respectively.*

End

3.2. Encryption

This section describes the encryption and decryption processes. It also provides the correctness of the encryption. The proposed encryption algorithm uses the public matrices $S, T \in M_n(G)$ and a secure $k-$bit hash function, $Hash$, i.e., the size of the output of $Hash(\cdot)$ is $k-$bit. Because of our construction of encryption, we will consider the plaintext m as a list of $t-$blocks, $m_0, m_1, \ldots, m_{t-1}$ each of length k-bit. If the size of the last block m_{t-1} is less than k, we can add some bits, i.e., padding, such that the size of m_{t-1} is $k-$bit.

The sender encrypts the message m by choosing two random circulant matrices, say B_1 and B_2, then the sender calculates $U = B_1 \boxtimes S \boxtimes B_2$ and $U' = B_1 \boxtimes T \boxtimes B_2$. Then the sender computes $H_0 = Hash(U')$, where $Hash$ is a secure hash function. If the plaintext m is a sequence of blocks, $m_0, m_1, \ldots, m_{t-1}$, then the sender computes $H_i = Hash(H_{i-1})$ and obtains the corresponding ciphertext block g_i by computing $g_i = m_i \oplus H_i$, see Figure 1. The ciphertext is $c = (U, (g_0, g_1, \ldots, g_{t-1}))$.

To decrypt the ciphertext, c, the receiver has the private-key (P_1 and P_2), so the receiver can easily compute $U' = P_1 \boxtimes U \boxtimes P_2$. Similarly, as in encryption, the receiver has $H_0 = Hash(U')$ and $H_i = Hash(H_{i-1})$. Since $g_i = m_i \oplus H_i$ using Algorithm 2, we have

$$g_i \oplus H_i = (m_i \oplus H_i) \oplus H_i = m_i.$$

This recovers the plaintext m; see Figure 2.

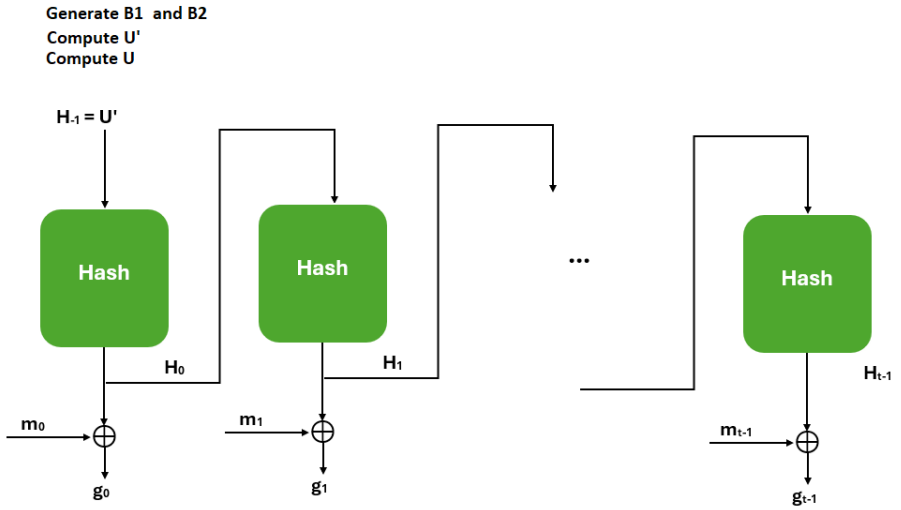

Figure 1. Encryption.

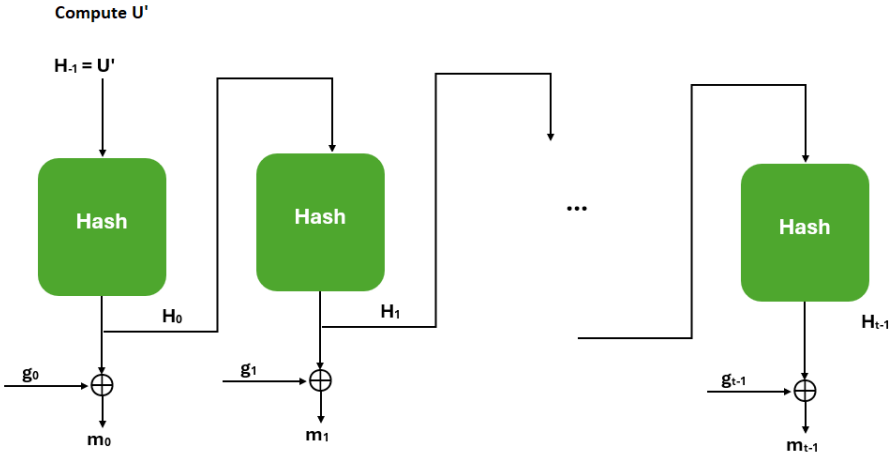

Figure 2. Decryption.

Algorithm 2. ***Encryption***
Input:

– *Public matrices* $S, T \in M_n(G)$.

– *A secure* $k-$*bit hash function Hash.*

– *A plaintext* $m = (m_0, \ldots, m_{t-1})$, $m_i \in \{0,1\}^k, 0 \leq i < t$.

Output: *ciphertext c.*
Begin

1: *Generate randomly two circulant matrices* $B_1, B_2 \in M_n(G)$.
2: $U' = B_1 \boxtimes T \boxtimes B_2$.
3: $U = B_1 \boxtimes S \boxtimes B_2$.
4: $H_{-1} = U'$.

5: ***for*** $i = 0$ *to* $t - 1$ ***do***
6: $H_i = Hash(H_{i-1})$.
7: $g_i = m_i \oplus H_i$.
8: ***end for***
9: ***Return*** *the ciphertext* $c = (U, (g_0, g_1, \ldots, g_{t-1}))$.

End

Let $c = Enc_e(m)$ denote the ciphertext c obtained by encrypting a message m using Algorithm 2, where e is the public-key.

Remark 3.1.
- If each element x_i of a matrix $X \in M_n(G)$ is represented by $b-$bits, then X can be considered as a sequence of bits of length $b * n^2$. Thus, we can easily apply a secure $k-$bit hash function, $Hash$, on X to produce a hash value of length k. Thus, Step 6 at $i = 0$ ($H_0 = Hash(H_{-1})$) in Algorithm 2 can be easily implemented.
- Both matrices B_1, and B_2 should be kept secret, and there is no necessity to retain them, i.e., destroy them.

Algorithm 3. ***Decryption***
Input:

– *Private matrices* $P_1, P_2 \in M_n(G)$.

– *Ciphertext* $c = (U, (g_0, g_1, \ldots, g_{t-1}))$, *where* $U \in M_n(G)$ *and* $g_i \in \{0,1\}^k, 0 \leq i < t$.

– *A secure* $k-$*bit hash function* $Hash$.

Output: *The plaintext* $m = (m_0, \ldots, m_{t-1})$, *where* $m_i \in \{0,1\}^k, 0 \leq i < t$.
Begin

1: $U' = P_1 \boxtimes U \boxtimes P_2$.
2: $H_{-1} = U'$.
3: ***for*** $i = 0$ *to* $t - 1$ ***do***
4: $H_i = Hash(H_{i-1})$.
5: $m_i = g_i \oplus H_i$.
6: ***end for***
7: ***Return*** *the plaintext* $m = (m_0, m_1, \ldots, m_{t-1})$

End

Similarly, let $m = Dec_d(c)$ denote the plaintext m recovered from the ciphertext c using Algorithm 3, where d is the private-key.

In Theorem 3.2, we prove the correctness of the proposed public-key encryption.

Theorem 3.2. *Let* $e = (S, T)$ *be the public-key of the proposed cryptosystem for encryption and* $d = (P_1, P_2)$ *be the corresponding private-key. Then* $m = Dec_d(Enc_e(m))$ *for every message* m.

Proof. Represent the message m as a sequence of blocks $m_i, 0 \leq i < t$, such that each block m_i is of size k, where k is the length of a hash value. This means that m can be written as $m = (m_0, \ldots, m_{t-1})$, $m_i \in \{0,1\}^k$. Using Algorithm 2, we get that $Enc_e(m) = c$, where $c = (U, (g_0, g_1, \ldots, g_{t-1}))$. We have that $U = B_1 \boxtimes S \boxtimes B_2$, for two random matrices $B_1, B_2 \in M_n(G)$. Therefore,

$$U' = P_1 \boxtimes U \boxtimes P_2 = P_1 \boxtimes B_1 \boxtimes S \boxtimes B_2 \boxtimes P_2.$$

Since P_1, P_2 and B_1, B_2 are circulant matrices, we have the following:

$$\begin{aligned}\text{(using Algorithm 3) } U' &= P_1 \boxtimes U \boxtimes P_2\\ &= P_1 \boxtimes B_1 \boxtimes S \boxtimes B_2 \boxtimes P_2\\ &= B_1 \boxtimes P_1 \boxtimes S \boxtimes P_2 \boxtimes B_2\\ &= B_1 \boxtimes T \boxtimes B_2\\ &= U' \text{ (using Algorithm 2)}\end{aligned}$$

Note that at Step 4 in Algorithms 2 and at Step 2 in Algorithm 3, we have $H_{-1} = U'$, and $H_i = Hash(H_{i-1})$ for $i = 0, 1, \ldots, t-1$. Since in Algorithm 2 we have $g_i = m_i \oplus H_i$, we also have, in Algorithm 3, that

$$g_i \oplus H_i = (m_i \oplus H_i) \oplus H_i = m_i.$$

□

4. Example

We give an example of the key generation and encryption process. Let $n = 5$. Using Algorithm 1, the generated keys are as follows.

The private-key is

$$P_1 = \begin{bmatrix} 38 & 55 & 247 & 209 & 19 \\ 19 & 38 & 55 & 247 & 209 \\ 209 & 19 & 38 & 55 & 247 \\ 247 & 209 & 19 & 38 & 55 \\ 55 & 247 & 209 & 19 & 38 \end{bmatrix}, \qquad P_2 = \begin{bmatrix} 128 & 198 & 219 & 164 & 91 \\ 91 & 128 & 198 & 219 & 164 \\ 164 & 91 & 128 & 198 & 219 \\ 219 & 164 & 91 & 128 & 198 \\ 198 & 219 & 164 & 91 & 128 \end{bmatrix}.$$

The public-key is

$$S = \begin{bmatrix} 149 & 6 & 84 & 46 & 9 \\ 244 & 46 & 171 & 241 & 221 \\ 17 & 180 & 42 & 226 & 248 \\ 216 & 246 & 45 & 60 & 0 \\ 207 & 176 & 190 & 36 & 34 \end{bmatrix}, \qquad T = \begin{bmatrix} 247 & 219 & 209 & 209 & 164 \\ 244 & 128 & 246 & 219 & 164 \\ 164 & 209 & 247 & 219 & 209 \\ 164 & 247 & 219 & 84 & 84 \\ 209 & 209 & 164 & 38 & 128 \end{bmatrix}.$$

Consider the plaintext m in the hexadecimal representation (of size 512 bits) as follows:

$$m = \begin{bmatrix} 35 & 0a & 6a & c1 & 6c & fe & 76 & 7c \\ a7 & 40 & b8 & 4d & 35 & 3c & 0d & 93 \\ 34 & 2c & 63 & fe & 79 & f3 & 42 & cf \\ 21 & 51 & 0a & f6 & 3e & 0a & 31 & 92 \\ 6c & d0 & ff & 28 & bf & 4a & a6 & 1e \\ 7a & 13 & 40 & c7 & 92 & 76 & 70 & 73 \\ 00 & da & c0 & c4 & 7e & 8d & 06 & f3 \\ f9 & 93 & 61 & a2 & e0 & ac & 17 & 08 \end{bmatrix}.$$

To encrypt m using Algorithm 2, two random circulant matrices B_1 and B_2 are generated:

$$B_1 = \begin{bmatrix} 120 & 148 & 144 & 171 & 250 \\ 250 & 120 & 148 & 144 & 171 \\ 171 & 250 & 120 & 148 & 144 \\ 144 & 171 & 250 & 120 & 148 \\ 148 & 144 & 171 & 250 & 120 \end{bmatrix}, \quad B_2 = \begin{bmatrix} 1 & 28 & 250 & 87 & 94 \\ 94 & 1 & 28 & 250 & 87 \\ 87 & 94 & 1 & 28 & 250 \\ 250 & 87 & 94 & 1 & 28 \\ 28 & 250 & 87 & 94 & 1 \end{bmatrix}.$$

Then, we compute $U' = B_1 \boxtimes T \boxtimes B_2$ and $U = B_1 \boxtimes S \boxtimes B_2$ as follows:

$$U' = \begin{bmatrix} 246 & 246 & 219 & 164 & 128 \\ 148 & 94 & 219 & 209 & 209 \\ 94 & 219 & 84 & 84 & 148 \\ 209 & 164 & 128 & 209 & 209 \\ 219 & 209 & 209 & 148 & 94 \end{bmatrix}, \quad U = \begin{bmatrix} 46 & 28 & 28 & 244 & 244 \\ 28 & 226 & 148 & 94 & 28 \\ 246 & 216 & 28 & 216 & 246 \\ 94 & 190 & 28 & 28 & 148 \\ 84 & 84 & 46 & 148 & 94 \end{bmatrix}.$$

The suggested hash function to be used is the Keccak SHA3-512 [21]. Therefore, we have $H_0 = Hash(U')$ in hexadecimal form as follows:

$$H_0 = \begin{bmatrix} 77 & 20 & F0 & 56 & C5 & 00 & 70 & B0 \\ F6 & 22 & 74 & AE & C2 & 7B & CD & 43 \\ 6A & C8 & B7 & 49 & 8B & 18 & 8D & D6 \\ AB & D1 & 6E & F8 & 6D & F8 & 48 & 3D \\ 20 & CE & 49 & F5 & 38 & 62 & AE & 57 \\ 45 & AF & 85 & 27 & 29 & F8 & 75 & FB \\ B2 & 60 & 42 & CD & D9 & 20 & FD & 8A \\ E5 & 33 & 82 & 5C & 24 & 23 & EE & 07 \end{bmatrix}.$$

Since the size of the given plaintext m equals the size of the hash digest of the Keccak SHA3-512, we consider the given plaintext as one block. Thus, the ciphertext of m is $c = (U, g_0)$, where $g_0 = m \oplus H_0$:

$$g_0 = \begin{bmatrix} 42 & 2a & 9a & 97 & a9 & fe & 06 & cc \\ 51 & 62 & cc & e3 & f7 & 47 & c0 & d0 \\ 5e & e4 & d4 & b7 & f2 & eb & cf & 19 \\ 8a & 80 & 64 & 0e & 53 & f2 & 79 & af \\ 4c & 1e & b6 & dd & 87 & 28 & 08 & 49 \\ 3f & bc & c5 & e0 & bb & 8e & 05 & 88 \\ b2 & ba & 82 & 09 & a7 & ad & fb & 79 \\ 1c & a0 & e3 & fe & c4 & 8f & f9 & 0f \end{bmatrix}.$$

Similarly, for decryption, we have the private-key P_1 and P_2, while the ciphertext is (U, g_0). We compute $U' = P_1 \boxtimes U \boxtimes P_2$ and $H_0 = Hash(U')$. Since, in this example, we consider the plaintext as one block, we have the plaintext $m = g_0 \oplus H_0$.

5. Security analysis

This section discusses the security of the proposed public-key encryption.

5.1. Brute-force attack

In this attack, the adversary finds all possible keys. Since the private matrices, P_1, and P_2 are circulant, knowing any row in the matrix leads to discovering the private matrix. If each element a_{ij} in a matrix $A \in M_n(G)$ is represented by $b-$bits, then the brute-force attack takes time $O(2^{bn})$ to find the matrix key. For example, if $b = 8$, then it is secure to take $n \geq 13$ to avoid the attack. If $A \in M_n(G)$ is not a circulant matrix, then the brute-force attack takes $O(2^{bn^2})$. For example, if $b = 8$, then it is safe to take $n \geq 4$ to avoid the attack. Note that, as b increases, n decreases, and vice versa. Therefore, estimating the private keys P_1, P_2, or the matrix U' (Step 2 in Algorithm 2 and Step 1 in Algorithm 3) is hard. Thus, the brute-force attack is not practical for the proposed encryption.

For small values of n and b, there is a (very) small possibility that the entries of matrices, such as T in Algorithm 1 and U' in Algorithm 2, are one or two repeated elements.

Thus, to prevent such a situation and to increase the security of generating keys and encrypting messages, we introduce a new terminology called *matrix weight* to ensure that the number of different elements in each row or column in some generated matrices, such as T in Algorithm 1 and U' in Algorithm 2, is not small.

Definition 5.1. *Let $A = (a_{ij}) \in M_n(G)$. The matrix weight of A, denoted by $\Omega(A)$, is the minimum number of distinct elements in each row and column. Formally,*

$$\Omega(A) = minimum\{\#\{a_{0j}, 0 \leq j < n\}, \cdots, \#\{a_{(n-1)j}, 0 \leq j < n\}, \#\{a_{i0}, 0 \leq i < n\}, \cdots, \#\{a_{i(n-1)}, 0 \leq i < n\}\}.$$

Note that the cardinality, denoted by #, of a set is the total number of unique elements in a set.

For example, consider the following matrix:

$$A = \begin{bmatrix} 2 & 4 & 5 & 3 \\ 1 & 6 & 2 & 1 \\ 4 & 3 & 2 & 3 \\ 5 & 2 & 1 & 3 \end{bmatrix},$$

we have $\Omega(A) = 2$.

Now, we give an example to show that there is a possibility that all elements of a matrix are equal.

Let the private-key be

$$P_1 = \begin{bmatrix} 21 & 173 & 96 & 32 & 57 \\ 57 & 21 & 173 & 96 & 32 \\ 32 & 57 & 21 & 173 & 96 \\ 96 & 32 & 57 & 21 & 173 \\ 173 & 96 & 32 & 57 & 21 \end{bmatrix}, \qquad P_2 = \begin{bmatrix} 26 & 190 & 11 & 236 & 178 \\ 178 & 26 & 190 & 11 & 236 \\ 236 & 178 & 26 & 190 & 11 \\ 11 & 236 & 178 & 26 & 190 \\ 190 & 11 & 236 & 178 & 26 \end{bmatrix}.$$

We have $\Omega(P_1) = 5$, while $\Omega(P_2) = 5$.

The public-key is

$$S = \begin{bmatrix} 75 & 52 & 232 & 56 & 179 \\ 53 & 113 & 99 & 101 & 229 \\ 9 & 253 & 249 & 70 & 103 \\ 199 & 234 & 96 & 11 & 59 \\ 238 & 75 & 213 & 227 & 116 \end{bmatrix}, \qquad T = \begin{bmatrix} 57 & 96 & 227 & 236 & 57 \\ 236 & 96 & 236 & 59 & 96 \\ 75 & 57 & 236 & 57 & 238 \\ 236 & 57 & 236 & 96 & 75 \\ 236 & 96 & 236 & 229 & 57 \end{bmatrix}.$$

We have $\Omega(S) = 5$, while $\Omega(T) = 2$.

The two random circulant matrices B_1, B_2 in Algorithm 2 are

$$B_1 = \begin{bmatrix} 63 & 221 & 52 & 170 & 222 \\ 222 & 63 & 221 & 52 & 170 \\ 170 & 222 & 63 & 221 & 52 \\ 52 & 170 & 222 & 63 & 221 \\ 221 & 52 & 170 & 222 & 63 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} 180 & 235 & 26 & 90 & 14 \\ 14 & 180 & 235 & 26 & 90 \\ 90 & 14 & 180 & 235 & 26 \\ 26 & 90 & 14 & 180 & 235 \\ 235 & 26 & 90 & 14 & 180 \end{bmatrix}.$$

We have $\Omega(B_1) = 5$, while $\Omega(B_2) = 5$,

$$U' = \begin{bmatrix} 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \\ 63 & 63 & 63 & 63 & 63 \end{bmatrix}.$$

We have $\Omega(U') = 1$.

Given a matrix, $A \in M_n(G)$, we propose that $\Omega(A) \geq \omega$, where ω is a security parameter referring to the minimum number of different elements needed in each row and column of A, i.e., $\Omega(A) \geq \omega$. Note that this does not mean that the matrix A has only ω different elements. We could not calculate the optimal value for ω.

In general, efficiency and security are inversely proportional. For different values of ω, we show, in Table 1, the number of matrices X with $\Omega(X) \geq \omega$ from 10^4 randomly generated public keys for different matrix dimensions. The results in the table show that large ω may lead to inefficiency of the encryption since we try generating different matrices until we find one with Ω greater than or equal to ω.

Table 1. The number of public-keys with $\Omega(T) \geq \omega$ per 10^4 randomly public-keys.

n	$\omega = 3$	$\omega = 4$	$\omega = 5$
15	8355	6474	7
30	9609	9232	392
48	9805	9673	3461
64	9887	9791	5016
80	9919	9919	6658

We claim that it is safe to set $\omega \geq 3$ since we do not know the position of different elements, and a matrix may have more than ω different elements.

Note that if n and b increase, the possibility of finding a matrix with a small weight decreases. Determining an optimal ω may not be easy.

Open Problem 5.2. *Given $M_n(G)$. Finding the optimal value of ω that renders estimating U infeasible.*

Thus, it is better to check whether the generated matrix T in Step 3 of Algorithm 1 has a weight less than ω or not. If $\Omega(T) < \omega$, then we have to generate other matrices, P_1, P_2, and so T. Similarly, for U' at Step 2 in Algorithm 2. We did not check the weight of P_1, P_2, B_1, and B_2 because they are random matrices.

Algorithm 4. ***Modified key genereation***
Input:

– *n: matrix dimension.*

– *ω: security parameter (matrix weight).*

Output: *Public matrices* $S, T \in M_n(G)$ *and private circulant matrices* $P_1, P_2 \in M_n(G)$.
Begin

1: *Generate randomly a matrix* $S \in M_n(G)$.
2: *Generate randomly two circulant matrices* $P_1, P_2 \in M_n(G)$.
3: *Compute* $T = P_1 \boxtimes S \boxtimes P_2$.
4: ***if*** $\Omega(T) < \omega$ ***then***
5: *Go to step 1.*
6: ***end if***
7: ***Return*** (S, T), *and* (P_1, P_2) *as the public-key and private-key respectively.*

End

Similarly, we can modify Algorithm 2.

5.2. Knowing U′

If an adversary obtains U', then he/she can recover the plaintext but not the private-key due to the hardness of Problems 2.4 and 2.5. On the other hand, knowing U' does not imply knowing the random matrices, B_1, and B_2, in Algorithm 2 because of Problem 2.4. If an adversary wants to estimate U', then he/she takes $O(b^{n^2})$, as mentioned in Section 5.1.

5.3. Known plaintext attack

Obtaining the matrix U' is hard due to Problem 2.5. However, if an adversary knows plaintext $m = m_0, m_1, \cdots, m_{t-1}$ and the corresponding ciphertext, $c = (U, (g_0, g_1, \ldots, g_{t-1}))$, then he/she can get $H_0 = g_0 \oplus m_0$. We have $H_0 = Hash(U')$ using Algorithm 2. Since $Hash$ is secure, the adversary cannot recover U'.

5.4. Chosen ciphertext attack

Assume that the challenge ciphertext is $c = (U, g)$. We investigate the possibility that an adversary with access to an oracle machine could return the plaintext of any suggested ciphertext $c' = (U', g')$ other than $c = (U, g)$. Note that, in the decryption process, the plaintext is obtained by performing the bitwise-XOR between g' and the hash value $Hash(U')$. Using a secure hash function, $Hash$, in the decryption makes the chosen ciphertext attack on the proposed encryption more difficult. However, choosing a ciphertext with a special pattern does not help the attacker obtain information about the private-key. Thus, the chosen ciphertext attack is inefficient.

5.5. Private-key recovery attack

The problem of finding the private-key P_1, P_2 from the public-key S, T is formulated as MAP (Problem 2.4). In [14], it is declared that MAP can be reduced to the NP-complete problem of solving

quadratic polynomial systems over the semiring $(G, \boxplus, \boxtimes)$. Currently, no method is known to solve the MAP in polynomial time. Therefore, this indicates that obtaining the private-key from the public-key is a hard problem.

5.6. Solving linear system

Otero et al. [22] proposed a method to find the maximal solution of a linear equation system defined over an additively idempotent semiring. Our proposed public-key encryption has the private-key (P_1 and P_2) and the public-key (S and T), where $T = P_1 \boxtimes S \boxtimes P_2$. If an attacker discovers either the matrix P_1 or P_2, then he/she can use the method of Otero et al. to obtain the other matrix. Thus, we must prevent the attacker from obtaining prior information about part of the private-key.

Durcheva and Danilchenko [18] presented Diffie–Hellman like key-exchange protocol based on tropical semiring and block matrices. Assuming that Alice and Bob are communicating, the key-exchange protocol proposed three public square tropical $(n \times n)$ matrices M, N, and T. The private-key of Alice is constructed by selecting an integer a and two private tropical polynomials of matrices p_1 and q_1 and substituting $A = p_1(M)$ and $B = q_1(N)$. The public attribute of Alice:

$$T_a = \sum_{i=0}^{a-1} A^{a-1-i} \boxtimes T \boxtimes B^i.$$

Similarly, the private-key of Bob contains an integer b and private tropical polynomials of matrices p_2 and q_2, where $C = p_2(M)$ and $D = q_2(N)$. The public attribute of Bob:

$$T_b = \sum_{i=0}^{b-1} C^{b-1-i} \boxtimes T \boxtimes D^i.$$

Otero et al. [22] succeeded in converting the cryptanalysis of the protocol proposed by Durcheva and Danilchenko [18] into a solution of a linear system based on the prior knowledge of the two public matrices M and N.

In our proposed public-key encryption, the private-key does not come from tropical polynomials of matrices. Therefore, the cryptanalysis of Otero et al. [22] is not applicable in the case of our proposed encryption. The security depends on the MAC and CMAC problems, which can be converted to 3-SAT.

Similarly, Alhussaini et al. [2] presented a cryptanalysis of the tropical stickel protocol proposed by Grigoriev and Shpilrain [7]. In this tropical stickel protocol, there are public matrices A, B, W, U. The relationship between U and W is based on private tropical polynomials of matrices p_1 and q_1 where $U = p_1(A) \boxtimes W \boxtimes q_1(B)$. But in our proposed public-key encryption, the public-key is two matrices, S and T, where $T = P_1 \boxtimes S \boxtimes P_2$ for two circulant private matrices P_1 and P_2. Thus, the attack in [2] does not work on our proposed encryption.

6. Experiments

This section provides the performance of key generation, encryption, and decryption processes for different matrix dimensions. The implementation is written in Python 3.7.8 and ran on an Intel(R) Core(TM) i5-8250U CPU 1.60 GHz under the Windows 10 operating system.

The implementation is byte-oriented; that is, the integer range is $[0, 255]$. The matrix dimensions we used in the implementation are $n = 15, 30, 48, 64, 80$. We have done two types of implementations:

(1) The first measures the performance of key generation, Algorithm 4. The second column in Table 2 shows the average running time (in seconds) to generate 10^3 random keys (public and private).
(2) The second measures the performance of the encryption and decryption processes. Since Algorithms 2 and 3 use a secure hash function, we used the Keccak SHA3-512 [21] implemented in Python ("hashlib" Module). The third and fourth columns in Table 2 show the average running times (in seconds) to encrypt plaintext and decrypt ciphertext, respectively. The number of tested blocks in encryption and decryption for each n is 2000, where each block is 512−bits. In Table 3, we calculate the size (in bytes) of the private and public keys of our encryption schema.

Table 2. The average execution time, in seconds, for generating keys, encryption, and decryption, where $\omega = 3$.

n	Key generation	Encryption	Decryption
15	0.011	0.0000	0.0000
30	0.075	0.0005	0.0001
48	0.305	0.0115	0.0061
64	0.741	0.0481	0.0250
80	1.40	0.2120	0.0720

Table 3. The size (in bytes) of private and public keys of the proposed encryption, where $\omega = 3$.

n	Size of private-key (P_1, P_2)	Size of public-key (S, T)
15	30	450
30	60	1800
48	96	4,608
64	128	8,192
80	160	12,800

Now, we compare our encryption schema with a recent public-key encryption [15]. Tables 2 and 3 show that the average running time for generating keys using our schema is less than that given in [15]. In addition, the average running times for encryption and decryption are shorter than those given in [15]. The drawback of the proposed encryption is the size of the public-key (not the private-key). It is longer than that given in [15].

The key generation of our schema is faster than the key construction in the key exchange protocols [14]. The size of the public-key in our schema is longer than that given in [14], but shorter than that given in [18].

7. Conclusions and future work

We have proposed a post-quantum public-key encryption. Our encryption scheme is robust and efficient for protecting the data, making it suitable for IoT platforms where devices often have limited resources and need to communicate quickly and safely. The encryption and decryption processes are faster than those given in [15]. The key generation algorithm is faster than that given in [14, 15, 18].

The main drawback of the proposed encryption is the size of the public-key. It is large compared to [15]. Although the security of the proposed cryptosystem depends on the hardness of solving MAP and CMAP, the use of a secure hash function prevents some attacks.

In the future, we intend to (1) find a method to reduce the size of the public-key, such as more efficient representations; (2) provide a formal security proof, in particular, for chosen plaintext and ciphertext attacks; (3) extend the cryptosystem to a digital signature.

Author contributions

Dieaa I. Nassr: Conceptualization, methodology, software, formal analysis, writing—original draft preparation, writing—review and editing; Hatem M. Bahig: Conceptualization, methodology, formal analysis, writing—original draft preparation, writing—review and editing; Mohamed A. G. Hazber: Writing—review and editing; Ibrahim M. Alseadoon: writing—review and editing; Hazem M. Bahig: Conceptualization, methodology, writing—original draft preparation, writing—review and editing, supervision, project administration. All authors have read and approved the final version of the manuscript for publication.

Use of Generative-AI tools declaration

The authors declare that they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors are grateful to the referees for their valuable comments that improved the manuscript. Also, the authors would like to acknowledge the support provided by the Scientific Research Deanship at the University of Ha'il—Saudi Arabia through project number RG-23 120.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. S. Alhussaini, S. Sergeev, On implementation of stickel's key exchange protocol over max-min and max-*t* semirings, *J. Math. Cryptol.*, **18** (2024), 20240014.
2. S. Alhussaini, S. Sergeev, Attacking tropical stickel protocol by MILP and heuristic optimization techniques, *IACR Cryptol. Eprint Arch.*, 2024, 1169.
3. I. Anshel, M. Anshel, D. Goldfeld, An algebraic method for public-key cryptography, *Math. Res. Lett.*, **6** (1999), 287–291.
4. J. Callas, L. Donnerhacke, H. Finney, R. Thayer, *OpenPGP message format*, 1998.

5. Z. Cao, The multi-dimension RSA and its low exponent security, *Sci. China Ser. E-Technol. Sci.*, **43** (2000), 349–354. https://doi.org/10.1007/BF02916982
6. W. Chen, J. Meng, The hardness of the closest vector problem with preprocessing over ℓ_∞ norm, *IEEE Trans. Inform. Theory*, **52** (2006), 4603–4606. https://doi.org/10.1109/TIT.2006.881835
7. D. Grigoriev, V. Shpilrain, Tropical cryptography, *Comm. Algebra*, **42** (2014), 2624–2632. https://doi.org/10.1080/00927872.2013.766827
8. T. Dierks, C. Allen, *RFC 2246: The TLS protocol version 1.0*, 1999. Available from: `http://www.ietf.org/rfc/rfc2246.txt`
9. W. Diffie, M. Hellman, New directions in cryptography, *IEEE Trans. Inform. Theory*, **22** (1976), 644–654. https://doi.org/10.1109/TIT.1976.1055638
10. M. Durcheva, *Semirings as building blocks in cryptography*, Cambridge Scholars Publishing, 2019.
11. T. El-Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inform. Theory*, **31** (1985), 469–472. https://doi.org/10.1109/TIT.1985.1057074
12. D. Hankerson, A. Menezes, Elliptic curve discrete logarithm problem, In: *Encyclopedia of cryptography and security*, Boston: Springer, 2011, 397–400. https://doi.org/10.1007/978-1-4419-5906-5_246
13. J. Hoffstein, J. Pipher, J. H. Silverman, An introduction to mathematical cryptography, In: *Undergraduate texts in mathematics*, New York: Springer, 2014. https://doi.org/10.1007/978-1-4939-1711-2
14. H. Huang, X. Jiang, C. Peng, G. Pan, A new semiring and its cryptographic applications, *AIMS Mathematics*, **9** (2024), 20677–20691. https://doi.org/10.3934/math.20241005
15. H. Huang, C. Li, L. Deng, Public-key cryptography based on tropical circular matrices, *Appl. Sci.*, **12** (2022), 7401. https://doi.org/10.3390/app12157401
16. M. Joye, *Security analysis of RSA type cryptosystem*, PhD thesis, Université catholique de Louvain, 1997.
17. K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. S. Kang, C. Park, New public-key cryptosystem using braid groups, In: *Lecture notes in computer science*, Heidelberg: Springer, **1880** (2000). https://doi.org/10.1007/3-540-44598-6_10
18. M. Durcheva, K. Danilchenko, Secure key exchange in tropical cryptography: Leveraging efficiency with advanced block matrix protocols, *Mathematics*, **12** (2024), 1429. https://doi.org/10.3390/math12101429
19. D. Micciancio, S. Goldwasser, Closest vector problem. In: *Complexity of lattice problems*, Boston: Springer, **671** (2002), 45–68. https://doi.org/10.1007/978-1-4615-0897-7_3
20. D. I. Nassr, M. Anwar, H. M. Bahig, *New public key cryptosystem*, Cryptology ePrint Archive, 2021.
21. National institute of standards and technology, *SHA-3 Standard: Permutation-based Hash and extendable-output functions: FiPS PUB 202*, 2015. Available from: `https://csrc.nist.gov/pubs/fips/202/final`

22. Á. O. Sánchez, D. C. Portela, J. A. López-Ramos, On the solutions of linear systems over additively idempotent semirings, *Mathematics*, **12** (2024), 2904. https://doi.org/10.3390/math12182904

23. R. L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Commun. ACM*, **21** (1978), 120–126. https://doi.org/10.1145/359340.359342

24. P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM Rev.*, **41** (1999), 303–332. https://doi.org/10.1137/S0036144598347011

25. V. Shpilrain, A. Ushakov, The conjugacy search problem in public key cryptography: Unnecessary and insufficient, *Appl. Algebra Engrg. Comm. Comput.*, **17** (2006), 285–289. https://doi.org/10.1007/s00200-006-0009-6

26. E. Stickel, A new method for exchanging secret keys, In: *Third international conference on information technology and applications (ICITA'05)*, Sydney: IEEE, 2005, 426–430. https://doi.org/10.1109/ICITA.2005.33

27. H. Vandiver, Note on a simple type of algebra in which the cancellation law of addition does not hold, *Bull. Am. Math. Soc.*, **40** (1934), 914–920.

28. T. Ylonen, C. Lonvick, *The secure shell (SSH) protocol architecture*, 2006.

Appendix

This appendix contains some data from [14, 18] to compare it with our results.

Table 4 presents the average execution time, in seconds, to generate keys for the protocol [14] using a processor faster than our processor used in the implementation. By comparing the data in Tables 2 and 4, we find that the key generation of our system is faster than the key generation of [14].

Table 4. The average execution time, in seconds, for generating keys [14].

n	Key generation
20	0.2066
25	0.4029
30	0.6872
35	1.1041
40	1.6526
45	2.3468

Table 5 presents the size of the private and public keys (in megabytes MB) [18]. By comparing Tables 3 and 5, we find that the key size of our schema is shorter than the key size given in [18].

Table 5. Private and public key size (in MB) [18].

n	Public-key	Private-key
60	27.3	27.3
65	32.3	32.3
70	37.5	37.5
75	43.0	43.0
80	48.8	48.8
85	54.8	54.8
90	61.0	61.0