Chapter 4

Symmetric encryption scheme based on quasigroup

The rise in the use of low memory and computing power devices such as smartphones, sensors and iPads (or tablets) has created a demand for encryption methods that are well suited for them. Furthermore, with the growing adoption of cloud services, the amount of data exchanged between these devices has increased significantly. As a result, the development of lightweight cryptographic primitives has attracted significant attention in recent years. Therefore, to address the needs of these environments, NIST initiated a project to seek, assess, and standardize lightweight cryptographic protocols. In 2023, The Ascon family [41] was formally selected by NIST for standardization of lightweight cryptography applications .

In 2012, Battey and Parakh proposed a block encryption scheme based on quasigroups [5], both with and without cipher block chaining (CBC). They evaluated their scheme's randomness (or entropy) by utilizing the NIST statistical test suite and concurrently compared the performance with AES-256.

In 2021, Tiwari et al. introduced a quasigroup based lightweight block cipher, namely *INRU*, [122]. They utilized a sequence of string transformations based on the structure of quasigroups in constructing the round functions for both encryption and decryption algorithms. They proved that their scheme was robust against various cryptanalytic attacks such as linear attack, algebraic attacks and the standard differential attack. Additionally, they performed a comprehensive statistical analysis using NIST test suite in various modes of operation, including CBC, CFB, OFB and CTR while comparing their results with AES-128 under the same configuration.

Similarly, in 2023, Chauhan et al. proposed an ultra lightweight block cipher by utilizing the string transformations based on quasigroup structure referred as *BCWST* [21]. They assessed the randomness of their scheme by analyzing the

output of NIST test suite [4] and avalanche criteria. Additionally, they proved that their design is secured against various cryptanalytic attacks like standard linear attack, standard differential attack, and algebraic attacks.

This motivates further research in designing an efficient quasigroup based symmetric encryption schemes which have lower complexity and provide provable security

In 1982, Goldwasser and Micali [114] gave the concept of provable security for public key cryptosystems. Subsequently, in 1997, Bellare et al. [7] adapted this concept to the symmetric setting. Prior to this, Shannon introduced the concept of perfect secrecy for cryptosystems, explained by Stinson [119]. In a perfectly secure cryptosystem, for any two distinct plaintexts P_1, P_2, and for given legitimate ciphertext C corresponding to any one of plaintexts, the probability of C being the result of encrypting either P_1 or P_2 is equally likely.

For the context of this chapter, primarily we discuss two notions of security for symmetric encryption scheme: chosen-plaintext attack (CPA) and chosen-ciphertext attack (CCA). To understand these security notions, we mainly focus on the conceptual framework discussed by Bellare et al. in [7], detailed in [69].

Consider a symmetric encryption scheme $SE = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ where $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ denoted the key generation, encryption and decryption algorithms respectively. Formal security notions for SE are discussed below:

Indistinguishability under chosen-plaintext attack (IND-CPA) and chosen-ciphertext attack (IND-CCA)

Let $b \in \{0, 1\}$ and $\mathtt{k} \in \mathbb{N}$. Suppose $\mathcal{A}_{cpa}$ represents an adversary that can access encryption oracle $\mathcal{E}_K(\cdot)$ only and $\mathcal{A}_{cca}$ is an adversary that can access both encryption $\mathcal{E}_K(\cdot)$ and decryption $\mathcal{D}_K(\cdot)$ oracles. Now we define the following experiments:

Algorithm 1 $\mathbf{Exp}^{ind-cpa-b}_{SE,\mathcal{A}_{cpa}}(\mathtt{k})$

$K \xleftarrow{R} \mathcal{K}(\mathtt{k})$
$(x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{E}_K(\cdot)}_{cpa}(\mathtt{k})$
$c \leftarrow \mathcal{E}_K(x_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{E}_K(\cdot)}_{cpa}$
Return b'

Algorithm 2 $\mathbf{Exp}^{ind-cca-b}_{SE,\mathcal{A}_{cca}}(\mathtt{k})$

$K \xleftarrow{R} \mathcal{K}(\mathtt{k})$
$(x_0, x_1) \leftarrow \mathcal{A}^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}_{cca}(\mathtt{k})$
$c \leftarrow \mathcal{E}_K(x_b)$
$b' \leftarrow \mathcal{A}^{\mathcal{E}_K(\cdot),\mathcal{D}_K(\cdot)}_{cca}$
Return b'

It is mandated that the two messages (x_0, x_1) queried to $\mathcal{E}_K(\cdot)$ always have equal length. Note that, $\mathcal{A}_{cca}$ cannot query $\mathcal{D}_K(\mathcal{E}_K(\cdot))$. Now we define the advantage of the adversaries via

$$\mathbf{Adv}^{ind-cpa}_{SE,\mathcal{A}_{cpa}}(\mathtt{k}) = Pr[\mathbf{Exp}^{ind-cpa-1}_{SE,\mathcal{A}_{cpa}} = 1] - Pr[\mathbf{Exp}^{ind-cpa-0}_{SE,\mathcal{A}_{cpa}} = 1]$$

$$\mathbf{Adv}^{ind-cca}_{SE,\mathcal{A}_{cca}}(\mathtt{k}) = Pr[\mathbf{Exp}^{ind-cca-1}_{SE,\mathcal{A}_{cca}} = 1] - Pr[\mathbf{Exp}^{ind-cca-0}_{SE,\mathcal{A}_{cca}} = 1]$$

The advantage functions of the scheme SE can be described as follows: For any integers $t, q_e, \mu_e, q_d, \mu_d$,

$$\mathbf{Adv}^{ind-cpa}_{SE}(\mathtt{k}, t, q_e, \mu_e) = \max_{\mathcal{A}_{cpa}}\{\mathbf{Adv}^{ind-cpa}_{SE}(\mathtt{k})\}$$
$$\mathbf{Adv}^{ind-cca}_{SE}(\mathtt{k}, t, q_e, \mu_e, q_d, \mu_d) = \max_{\mathcal{A}_{cca}}\{\mathbf{Adv}^{ind-cca}_{SE}(\mathtt{k})\}$$

where the maximum is taken over all potential adversaries $\mathcal{A}_{cpa}$ and $\mathcal{A}_{cca}$ with "time complexity (t)". Each adversary is constrained to making at most q_e queries to the $\mathcal{E}_K(\cdot)$ oracle, resulting in μ_e ciphertexts that both adversaries observe in response to these encryption oracle queries. Additionally, in case of A_{cca}, making at most q_d queries to the $\mathcal{D}_K(\cdot)$ oracle, and the amount of plaintext they sees in response to decryption oracle queries are μ_d. The scheme SE is said to be IND-CPA secure (resp. IND-CCA secure) if the function $\mathbf{Adv}^{ind-cpa}_{SE}(\cdot)$ (resp. $\mathbf{Adv}^{ind-cca}_{SE}(\cdot)$) is negligible for any adversary $\mathcal{A}$ whose time complexity is polynomial in $\mathtt{k}$.

Left-or-Right indistinguishability under chosen plaintext attack (LOR-CPA) and chosen ciphertext attack (LOR-CCA)

Let $b \in \{0,1\}$ and $\mathtt{k} \in \mathbb{N}$. Let $\mathcal{A}_{cpa}$ be an adversary that has access to the oracle $\mathcal{E}_K(LR(\cdot,\cdot,b))$ and let A_{cca} be an adversary that has access to the oracles $\mathcal{E}_K(LR(\cdot,\cdot,b))$ and $\mathcal{D}_K(\cdot)$. Now we consider the following experiment:

Algorithm 3 $\mathbf{Exp}^{lor-cpa-b}_{SE,\mathcal{A}_{cpa}}(\mathtt{k})$

$K \xleftarrow{R} \mathcal{K}(\mathtt{k})$
$d \leftarrow \mathcal{A}^{\mathcal{E}_K(LR(\cdot,\cdot,b))}_{cpa}(\mathtt{k})$
Return d.

Algorithm 4 $\mathbf{Exp}^{lor-cca-b}_{SE,\mathcal{A}_{cca}}(\mathtt{k})$

$K \xleftarrow{R} \mathcal{K}(\mathtt{k})$
$d \leftarrow \mathcal{A}^{\mathcal{E}_K(LR(\cdot,\cdot,b)),\mathcal{D}_K(\cdot)}_{cca}(\mathtt{k})$
Return d.

It is mandated that the Adversary $\mathcal{A}$ queries two messages of equal length to $\mathcal{E}_K(LR(\cdot,\cdot,b))$. Note that A_{cca} cannot query $\mathcal{D}_K(\cdot)$ on the same ciphertext c output of the $\mathcal{E}_K(LR(\cdot,\cdot,b))$. Now we define the advantage of the adversaries via

$$\mathbf{Adv}^{lor-cpa}_{SE,\mathcal{A}_{cpa}}(\mathtt{k}) = Pr[\mathbf{Exp}^{lor-cpa-1}_{SE,\mathcal{A}_{cpa}} = 1] - Pr[\mathbf{Exp}^{lor-cpa-0}_{SE,\mathcal{A}_{cpa}} = 1]$$
$$\mathbf{Adv}^{lor-cca}_{SE,\mathcal{A}_{cca}}(\mathtt{k}) = Pr[\mathbf{Exp}^{lor-cca-1}_{SE,\mathcal{A}_{cca}} = 1] - Pr[\mathbf{Exp}^{lor-cca-0}_{SE,\mathcal{A}_{cca}} = 1]$$

The advantage function of the schemes are as follows: For any integers $t, q_e, \mu_e, q_d, \mu_d$,

$$\mathbf{Adv}^{lor-cpa}_{SE}(\mathtt{k}, t, q_e, \mu_e) = \max_{\mathcal{A}_{cpa}}\{\mathbf{Adv}^{lor-cpa}_{SE}(\mathtt{k})\}$$

$$\mathbf{Adv}_{SE}^{lor-cca}(\mathtt{k}, t, q_e, \mu_e, q_d, \mu_d) = \max_{\mathcal{A}_{cca}}\{\mathbf{Adv}_{SE}^{lor-cca}(\mathtt{k})\}$$

where the maximum is taken over all potential adversaries $\mathcal{A}_{cpa}$ and $\mathcal{A}_{cca}$ with "time complexity (t)". Each adversary is limited to making at most q_e queries to the $\mathcal{E}_K(LR(\cdot,\cdot,b))$ oracle, which results in amount of ciphertexts μ_e that both adversaries see in response to these encryption oracle queries. Additionally, in case of $\mathcal{A}_{cca}$, making at most q_d queries to the $\mathcal{D}_K(\cdot)$ oracle, and the amount of plaintext receives in response to its decryption oracle queries are μ_d. The scheme SE is said to be LOR-CPA secure (resp. LOR-CCA secure) if the function $\mathbf{Adv}_{SE}^{lor-cpa}(\cdot)$ (resp. $\mathbf{Adv}_{SE}^{lor-cca}(\cdot)$) is negligible for any adversary whose time complexity is polynomial in $\mathtt{k}$.

For practical purposes, block ciphers are used in various modes to provide confidentiality and authenticity. We briefly describe the different mode of operations in the following section.

Mode of operations

Several modes of operations have been proposed after realizing the practicality of block ciphers and these modes are divided mainly into three different categories: confidentiality modes, authenticated modes and authenticated-encryption modes. Some examples of confidentiality and authenticated modes of operations are: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), Counter (CTR), Cipher Block Chaining-Message Authentication Codes (CBC-MACs), Hash Message Authentication Code (HMAC) and Galois Message Authentication Code (GMAC). The authenticated-encryption mode of operation ensures both the confidentiality and authenticity of the ciphertext. Examples of the authenticated encryption schemes include: Galois Counter Mode (GCM) [109] and Counter with Cipher Block Chaining-Message Authentication Code [109]. For a detailed survey on block ciphers using different modes of operation, readers may refer [109].

In literature some confidentiality mode of operations has proven to be provable secure against chosen plaintext and chosen ciphertext attack. We discuss some of them along with their advantages (**Adv**).

- **Electronic code book (ECB)**: In FIPS 1981 [52], various modes of operation for Data Encryption Standards (DES) are described and ECB is one of them. It is a deterministic mode of operation, in which the blocks of messages are encrypted independently. This vulnerability makes ECB susceptible to chosen plaintext attack (CPA).

The lack of diffusion in ciphertext is the main drawback for this mode of operation. ECB encryption scheme transforms identical plaintext blocks into the identical ciphertext blocks and fails to effectively conceal the data patterns. For instance, it is easy to differentiate the encryption of a plaintext containing two identical blocks from the encryption of a plaintext that consists of two distinct blocks.

- **Cipher block chaining (CBC)** : In 1976, Ehrsam et al. introduced the CBC mode of operation [45]. In this mode, unlike ECB, an encrypted form of initialization vector is XORed with the first block of plaintext, and then each consecutive plaintext block is XORed with the preceding ciphertext block before the encryption process. The initialization vector guarantees the uniqueness of ciphertext when same message is encrypted more than once. In 1997, Bellare et al. [7] did a comprehensive security analysis for the CBC scheme and proved that it is LOR-CPA secure.

  The primary drawbacks include that the encryption scheme is sequential, and before encryption user must ensure that the message is padded to a scaler multiple of the cipher block size. Additionally, in the CBC mode, even a single bit change in the plaintext or initial vector can impact all the subsequent ciphertext blocks.

  In [7], Bellare et al. gave lower and upper bound on advantage (**Adv**) of CBC scheme using the *random permutations*, i.e. for $R = \{0,1\}^l$, $\mu_e \leq l \cdot 2^{1/2}$ and $q_e = \mu_e/l$

$$0.316\left(\frac{\mu_e^2}{l^2} - \frac{\mu_e}{l}\right) \cdot \frac{l}{2^l} \leq \mathbf{Adv}^{lor-cpa}_{CBC[R]}(\cdot, t, q_e, \mu_e) \leq \left(\frac{\mu_e^2}{l^2} - \frac{\mu_e}{l}\right) \cdot \frac{l}{2^l}.$$

  In [7], Bellare et al. also gave an upper bound on the advantage **Adv** of CBC scheme compared to the *pseudorandom permutations (prp)*. Suppose F belongs to pseudorandom permutations family with length l. For any t, q_e and $\mu_e = ql$,

$$\mathbf{Adv}^{lor-cpa}_{CBC[F]}(\cdot, t, q_e, \mu_e) \leq 2 \cdot \mathbf{Adv}^{prp}_{F}(t, q) + \frac{q^2}{2^{l+1}} + \left(\frac{\mu_e^2}{l^2} - \frac{\mu_e}{l}\right) \cdot \frac{1}{2^l}.$$

  Proposition 4.0.1. [63] *Consider a CBC scheme* $SE = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ *with random initial vector* IV *and* $\mathcal{E} : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *is an encryption map. Then*

$$\mathbf{Adv}^{ind-cca}_{SE}(t, 1, n, 1, 2n) = 1$$

for $t = \mathcal{O}(n)$*, in addition to the time required for a single application of*

F, the advantage of this adversary is 1 despite utilizing minimal resources: merely one query to each oracle. This unequivocally indicates that the scheme is IND-CCA insecure.

- **Cipher feedback block (CFB)**: In 2002, Alkassar et al. introduced the CFB mode of operation [2]. In CFB scheme, the first block of plaintext is XORed with the encrypted form of initial vector IV, while the succeeding blocks of plaintext are XORed with the encrypted form of the previous block cipher. NIST SP800-38A establishes the definition of CFB with a specified bit-width [44]. CFB mode requires a parameter, denoted by s, where $1 \leq s \leq b$ and b is block length. In the description of the CFB mode, each plaintext and ciphertext segment is s bits long. Sometimes, the value of s is also incorporated into the mode's nomenclature, such as CFB mode, CFB-8 mode, CFB-64 mode or CFB-128 mode of operation.

  The disadvantages of CFB are identical to those of the CBC scheme. In [128], Wooding gave the bound on advantage of CFB scheme using the pseudorandom function F. For arbitrary t_e, q_e and μ_e,

$$\mathbf{Adv}^{lor-cpa}(CFB, t_e, q_e, \mu_e) \leq 2 \cdot \mathbf{Adv}^{prf}(F, t_e + qt_f, q) + \frac{q(q-1)}{2^l}$$

  where $q = \lfloor (\mu_e + q_e(t-1))/t \rfloor + n_e$, t_f is some small constant, l is the length of message block and prf represent pseudorandom function family.

- **Output Feedback block (OFB)**: The OFB mode of operation is seen as another variant of the CBC. In OFB mode [109], first block of the plaintext is XORed with the encrypted form of initial vector and for the second block cipher, encrypted form of initial vector is passed to the second encryption block and then the second plaintext block XORed with the output of the encryption block.

  OFB mode achieves CPA security when the encryption function F is a pseudorandom function. This mode, while requiring sequential encryption, offers the advantage over CBC mode that most of the computation, specifically the computation of the pseudorandom stream, can be performed independently, irrespective of the specific message intended to be encrypted.

  In [128], Wooding gave the bound on advantage of OFB scheme using the pseudorandom function F. For any t_e, q_e and μ_e,

$$\mathbf{Adv}^{lor-cpa}(OFB, t_e, q_e, \mu_e) \leq 2 \cdot \mathbf{Adv}^{prf}(F, t_e + qt_f, q) + \frac{q(q-1)}{2^l}$$

where $q = \lfloor(\mu_e + q_e(t-1))/t\rfloor + n_e$, t_f is some small constant, l is the length of message block and prf represent pseudorandom function family.

- **Counter (CTR)**: In 1979, Diffie and Hellman introduced the CTR mode of operation. This mode is straightforward counter-based implementation of a block cipher. With each encryption operation, a counter-initiated value is XORed with the plaintext and then performs the encryption process to produce the ciphertext blocks. The use of a counter value for each block helps avoid establishing a relationship between plaintext and ciphertext.

  The main disadvantage of CTR mode is its dependence on a synchronous counter during both encryption and decryption process. The recovery of plaintext is erroneous when synchronization is lost.

  In [7], Bellare et al. gave the bound on advantage of CTR mode using the pseudorandom function F with input length l and output length L. For any t, q_e and $\mu_e = \min(q'L, L2^l)$,

$$\mathbf{Adv}^{lor-cpa}_{CTR[F]}(\cdot, t, q_e, \mu_e) \leq 2 \cdot \mathbf{Adv}^{prf}_{F}(t, q')$$

  Proposition 4.0.2. [63] *Consider a counter based encryption scheme $SE = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ with random counter CTR. Let $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be the corresponding family of functions. Then the advantage function of the SE is:*

$$\mathbf{Adv}^{ind-cca}_{SE}(t, 1, l, 1, n + l) = 1.$$

  As we can observe that the advantage of the adversary is 1 using just one query to each encryption and decryption oracle with the time complexity as $t = \mathcal{O}(n+l)$ plus one operation of F. This clearly represents that the counter based encryption scheme SE with random CTR is IND-CCA insecure.

In this chapter we focus on the construction of symmetric encryption scheme based on quasigroup, referred as SEBQ, using chained mode of operation and its security analysis. First, in Section 4.1 we give the preliminaries of Latin squares, number of Latin squares of fixed order and the string transformations based on structure of quasigroups. Section 4.2 is dedicated to the construction of SEBQ based on quasigroups using chained like mode of operations in which instead of ciphertext transformed random vector will be passed to next block for the encryption of succeeding blocks of plaintext. Section 4.3 is committed for the security analysis of SEBQ scheme, in which we prove the proposed encryption scheme is indistinguishable against chosen plaintext attack and achieve IND-CCA2 security

by utilizing the Fiestel transformation. We assess the randomness of the ciphertext by analyzing the output of NIST-test suite and avalanche criteria. In Section 4.4, we determine the computational complexity of the SEBQ scheme and provide the order of Latin squares to achieve 128-bit and 256-bit security against known ciphertext attack.

4.1 Enumeration of Latin squares and a string transformation based on quasigroups

Definition 4.1.1. [33] A Latin square is an array of size $n \times n$ filled with n distinct symbols, ensuring that each symbol appears precisely once in each row and once in each column.

Every quasigroup $(Q = \{x_1, x_2, \ldots, x_n\}, *)$ can be represented by a Latin square L of size n^2. For example, for the set $Q = \{1, 2, 3, 4, 5\}$, a Latin square and its corresponding parastrophe's Latin square are represented in Table 4.1:

$*$	1	2	3	4	5
1	1	2	3	4	5
2	2	1	4	5	3
3	3	5	1	2	4
4	4	3	5	1	2
5	5	4	2	3	1

$\backslash$	1	2	3	4	5
1	1	2	3	4	5
2	2	1	5	3	4
3	3	4	1	5	2
4	4	5	2	1	3
5	5	3	4	2	1

Table 4.1: Latin squares corresponding to quasigroup $(Q, *)$ and its parastrophes $(Q, \backslash)$

Definition 4.1.2. [84] Consider a matrix $A = (a_{i,j})$ of order $n \times n$. Then, the permanent of A is defined as:

$$perm(A) = \sum_{\tau \in S_n} \prod_{j=1}^{n} a_{j,\tau(j)}.$$

The summation is running for all τ belongs to symmetric group S_n

The cardinality of Latin square plays a significant role in theoretical security of quasigroup-based cryptographic protocols. However, calculating the number of Latin square [115] poses a challenge because there is exponential growth in the number of terms involved. Consequently, extensive research efforts have been dedicated to estimating the number of Latin square for specific order by employing

methods like volunteer counting [47, 68, 73, 123] or utilized specialized software like Singular/ SageMath [3, 123].

In 1992, Shao and Wei [115] determined a formula to calculate the number of Latin squares. Suppose $\tau_0(A)$ represents the total number of zeroes in matrix A and $perm(A)$ is the permanent of A defined as (4.1.2), the respective formula is described below:

Theorem 4.1.3. *Let $B_n = \{(b_{i,j})_{n\times n} \mid b_{i,j} \in \{0,1\}\}$ be the set of all $n \times n$ sized matrices. Then number of Latin squares is:*

$$L(n) = n! \sum_{A\in B_n} (-1)^{\tau_0(A)} \binom{perm(A)}{n} \tag{4.1}$$

Remark 4.1.4. Let $B_{k\times n} = \{(b_{i,j})_{k\times n} \mid b_{i,j} \in \{0,1\}\}$ be the set of $k \times n$ matrices. Then the formula to calculate the number of Latin rectangles is:

$$L(k \times n) = n! \sum_{A\in B_{k\times n}} (-1)^{\tau_0(A)} \binom{perm(A)}{n} \quad \text{for } k \leq n \tag{4.2}$$

Above formulae (4.1) and (4.2) are not easy for computation, however Dénes and Keedwell [33] used them to compute bounds on number of Latin squares, i.e.

$$\prod_{j=1}^{n} (j!)^{n/j} \geq L(n) \geq \frac{(n!)^{2n}}{n^{n^2}}. \tag{4.3}$$

For specific values of $n = 2^l$, where $l = 7, 8$, the approximate number of $L(n)$ is estimated as:

$$0.164 \times 10^{21091} \geq L(128) \geq 0.337 \times 10^{20666}$$
$$0.753 \times 10^{102805} \geq L(256) \geq 0.304 \times 10^{101724}$$

We give a table contains exact values of Latin square $L(n)$ with n-number of symbols.

In order to set-up an encryption scheme, we make use of the quasigroup $(Q, *)$ based transformations on $Q^l \times Q^n$. This is formally defined below:

Definition 4.1.5. [59]**(𝔢-transformation)** Consider a quasigroup $(Q, *) = (\mathbb{F}_2^k, *)$. For a given $b \in Q$, we define $F_b : Q \to Q$ as $F_b(a) = b * a$. Also, we define $f : Q^n \to Q^n$ as $f(b_1 b_2 \dots b_n) = d_1 d_2 \dots d_n$, where each $d_i = b_i$ for $i = \overline{1, n-1}$ and $d_n = b_1 + b_2 + \dots + b_n$ (where $\overline{1, n-1}$ represents integers ranging from 1 to $n-1$). Using these functions, we define the following:

n	Number of Latin squares of size n
1	1
2	2
3	12
4	576
5	161,280
6	812,851,200
7	61,479,419,904,000
8	108,776,032,459,082,956,800
9	5,524,751,496,156,892,842,531,225,600
10	9,982,437,658,213,039,871,725,064,756,920,320,000

Table 4.2: Number of Latin squares of size n

(*i*) For a given $\beta = b_1b_2\dots b_n \in Q^n$, we define a map $F_\beta : Q \to Q$ as $F_\beta = F_{b_n} \circ F_{b_{n-1}} \circ \dots \circ F_{b_1}$.

(*ii*) For a given $a \in Q$, we define a map $f_a^n : Q^n \to Q^n$ as $f_a^n(b_1b_2\dots b_n) = d_1d_2\dots d_n$, where each component maps differently as $d_i = F_{b_i} \circ F_{b_{i-1}} \circ \dots \circ F_{b_1}(a)$, $i = \overline{1,n}$; additionally, we define $F_a^n : Q^n \to Q^n$ as $F_a^n = f \circ f_a^n$.

(*iii*) Finally, we define $\mathfrak{e}$-transformation $F^{l,n} : Q^l \times Q^n \to Q^l \times Q^n$ as $F^{l,n}(a_1a_2\dots a_l,\ b_1b_2\dots b_n) = (c_1c_2\dots c_l,\ d_1d_2\dots d_n)$, where each component is computed as:

$$c_i = F_{\delta_{i-1}}(a_i),\ \delta_i = F_{a_i}^n(\delta_{i-1}) \text{ for } i = \overline{1,l}$$

assuming $\delta_0 = b_1b_2\dots b_n$, and $d_1d_2\dots d_n = \delta_l$.

Definition 4.1.6. [59] **($\mathfrak{d}$-transformation)** Consider a quasigroup $(Q,*) = (\mathbb{F}_2^k,*)$ and its corresponding parastrophe $\backslash$. For a given $b \in Q$, we define $G_b : Q \to Q$ as $G_b(c) = b\backslash c$. Also, we define $f : Q^n \to Q^n$ as $f(b_1b_2\dots b_n) = d_1d_2\dots d_m$ where each $d_i = b_i$ for $i = \overline{1,n-1}$ and $d_n = b_1 + b_2 + \dots + b_n$. Using these functions, we define the following:

(*i*) For a given $\beta = b_1b_2\dots b_n \in Q^n$, we define a map $G_\beta : Q \to Q$ as $G_\beta = G_{b_1} \circ G_{b_2} \circ \dots \circ G_{b_n}$.

(*ii*) For a given $c \in Q$, we define a map $g_c^n : Q^n \to Q^n$ as $g_c^n(b_1b_2\dots b_n) = d_1d_2\dots d_m$ and each component map differently as $d_{n-i+1} = G_{b_n} \circ G_{b_{n-1}} \circ \dots \circ G_{b_{n-i+1}}(c)$, $i = \overline{1,n}$; additionally, we define $G_c^n : Q^n \to Q^n$ as $G_c^n = f \circ g_c^n$.

(*iii*) Finally, we define $\mathfrak{d}$-transformation $G^{l,n} : Q^l \times Q^n \to Q^l \times Q^n$ as $G^{l,n}(c_1c_2\dots c_l, b_1b_2\dots b_n) = (a_1a_2\dots a_l, d_1d_2\dots d_n)$ where each component

computed as:

$$a_i = G_{\delta_{i-1}}(c_i),\ \ \delta_i = G^n_{c_i}(\delta_{i-1})$$

assuming $\delta_0 = b_1 b_2 \dots b_n$ and $d_1 d_2 \dots d_n = \delta_l$.

Theorem 4.1.7. [59] *From the functions defined in Definition 4.1.5 and Definition 4.1.6, we have the following:*

(*i*) $G^l_\beta(F^l_\beta(\alpha)) = \alpha$ *and* $F^l_\beta(G^l_\beta(\gamma)) = \gamma$,

(*ii*) F^l_β *and* G^l_β *are permutations of* Q^l.

The primary objective is to construct a symmetric encryption scheme $SEBQ = \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ that comprises of three algorithms: Key generation process ($\mathcal{K}$), Encryption process ($\mathcal{E}$) and Decryption process ($\mathcal{D}$).

4.2 Symmetric encryption scheme based on quasigroup

Consider a set $Q = \mathbb{F}_2^k$ for some positive integer k. Now, $a \in Q$ implies a can be written as k-tuple of elements from set $\{0, 1\}$. In the SEBQ scheme, three algorithms have been put forth, including *Key Generation algorithm*, an *Encryption algorithm* and the *Decryption algorithm*. The security of SEBQ depends on the number of quasigroups of particular order.

4.2.1 Key generation process

Generate a random Latin square of size $2^k \times 2^k$ by giving an input a parameter $k \in \mathbb{N}$, utilizing an algorithm proposed by Artamonov et al. [3]. The generated Latin square is being treated as a secret key for SEBQ scheme and the dimension of the key space is $|L(n)|$ for $n = 2^k$ where as $|L(n)|$ can be calculated by utilizing (4.1). To compute the parastrophe of the corresponding Latin square, we utilize (1.1), which states that the knowledge of $(Q, *)$ or combinatorial equivalent Latin square is equivalent to the knowledge of $(Q, \backslash)$.

4.2.2 Encryption process

Consider $M = (m_1, m_2, \dots, m_l) \in (\mathbb{F}_2^k)^l$, where each $m_i \in \mathbb{F}_2^k$, i.e. each block m_i is k-bit long and M is kl-bit long plaintext. In order to begin the encryption process, user needs to consider an initial vector $R^{(1)} = r_1^{(1)}, r_2^{(1)}, \dots, r_n^{(1)}$, i.e. a kn-bit long string, where each $r_i^{(1)} \in Q$.

Now, the encryption scheme proceeds by making use of $\mathfrak{e}$-transformation as defined in Definition 4.1.5. Define the intermediate map $E_* : Q \times Q^n \to Q \times Q^n$ as

$$E_*(m, (r_1, r_2, \ldots, r_n)) = (r_n * (r_{n-1} * (\cdots * (r_1 * m) \cdots)), (r'_1, r'_2, \ldots, r'_n)) \,. \quad (4.4)$$

Here, $(r'_1, r'_2, \ldots, r'_n) = F_m^n(r_1, r_2, \ldots, r_n)$. For the pictorial representation, refer Figure 4.1.

The encryption process runs sequentially by first applying E_* on m_1 using the initial vector. Subsequently, the initial vector gets updated and using this together with m_2, the E_* results in second block of ciphertext. Similarly, the initial vector is updated and utilized to encrypt m_i till the complete message is encrypted. We say that the ciphertext corresponding to M is given by the vector $(E_*(m_1), E_*(m_2), \ldots, E_*(m_l))$, where by $E_*(m_i)$ we consider only the first component. This process is explained in Algorithm 5.

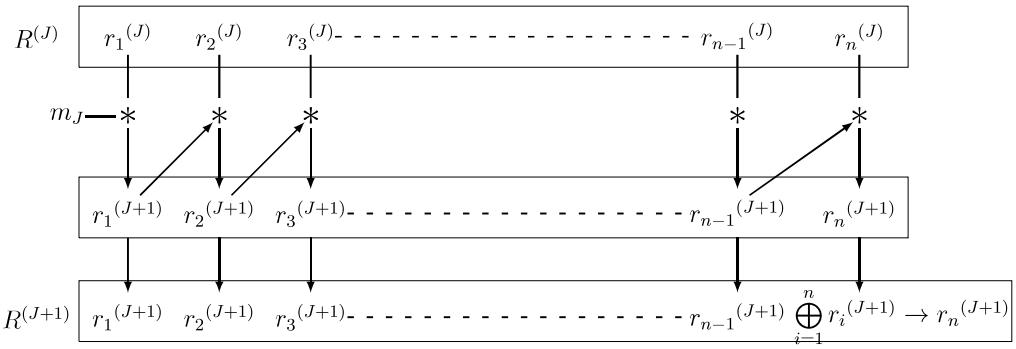

Figure 4.1: Transformation of initial vector for encryption function

To encrypt the i^{th} block of message M we need two inputs, i.e. a message block m_i and a random vector $R^{(i)}$, after encrypting the message block m_i we get output $(c_i, R^{(i+1)})$. For the pictorial representation of encryption scheme refer Figure 4.2 and Algorithm 5.

Algorithm 5 Encryption algorithm

Input: An $n \times n$ sized Latin square $*$ over Q, an initial (or, leader) random vector $R = (r_1, r_2, \ldots, r_n) \in Q^n$, and a message $M = (m_1, m_2, \ldots, m_l) \in Q^l = \mathbb{F}_2^{kl}$.

1: **for** $j = 1$ to l **do**
2: $\quad k_{0,j} \leftarrow m_j$ ▷ Assign $(k_{0,1}, k_{0,2}, \ldots, k_{0,l}) = (m_1, m_2, \ldots, m_l)$
3: **end for**
4: **for** $i = 1$ to n **do**
5: $\quad k_{i,0} \leftarrow r_i$ ▷ Assign $(k_{1,0}, k_{2,0}, \ldots, k_{n,0}) = (r_1, r_2, \ldots, r_n)$
6: **end for**
7: **for** $j = 1$ to l **do**
8: $\quad$ **for** $i = 1$ to n **do**
9: $\quad\quad k_{i,j} \leftarrow k_{i,j-1} * k_{i-1,j}$ ▷ Compute $k_{1,j}, k_{2,j}, \ldots, k_{n,j}$
10: $\quad$ **end for**
11: $\quad c_j \leftarrow k_{n,j}$ ▷ Compute the ciphertext block c_j as $k_{n,j}$
12: $\quad k_{n,j} \leftarrow k_{1,j} \oplus k_{2,j} \oplus \cdots \oplus k_{n,j}$ ▷ Update the value of $k_{n,j}$
13: **end for**

Output: $C = (c_1, c_2, \ldots, c_l) \in Q^l = \mathbb{F}_2^{kl}$

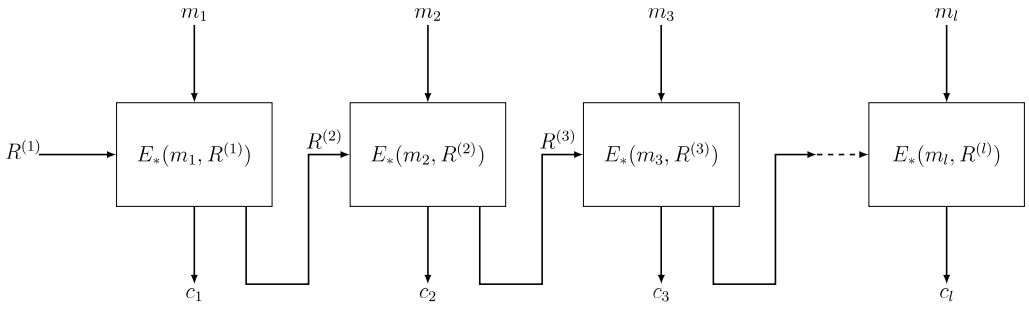

Figure 4.2: Encryption algorithm

Here, the encryption function, i.e. a function from $Q^l \to Q^l$, can be decomposed into l maps E_* each from $Q \times Q^n \to Q \times Q^n$ and same structure follows for the decryption function as well operation $*$ being replaced with $\backslash$.

4.2.3 Decryption process

The decryption process takes input a ciphertext $C = (c_1, c_2, \ldots, c_l) \in \mathbb{F}_2^{kl}$ along with the initial vector $R^{(1)} = r_1^{(1)}, r_2^{(1)}, \ldots, r_n^{(1)} \in Q^n$. The algorithm requires the multiplication table of the parastrophe $\backslash$ of quasigroup $(Q, *)$ which acts as the secret key.

For the decryption scheme, we use $\mathfrak{d}$-transformation of quasigroup using Definition 4.1.6. Define a map $D_\backslash : Q \times Q^n \to Q \times Q^n$ as $D_\backslash(c, (s_1, s_2, \ldots, s_n)) = (r_1\backslash(r_2\backslash(\ldots\backslash(r_n\backslash c)\ldots)), (s'_1, s'_2, \ldots, s'_n))$. Here, $(s'_1, s'_2, \ldots, s'_n) = G_c^n(s_1, s_2, \ldots, s_n)$.

Using same definition, transform the vector S after the decryption of each block of ciphertext given the initial vector $S^{(1)} = R^{(1)} = (r_1^{(1)}, r_2^{(1)}, \ldots, r_n^{(1)})$.

Algorithm 6 Decryption algorithm

Input: An $n \times n$ sized multiplication table of parastrophe $\backslash$ of Latin square $*$ over Q, an initial (or, leader) random vector $R = (r_1, r_2, \ldots, r_n) \in Q^n$, and a ciphertext $C = (c_1, c_2, \ldots, c_l) \in Q^l = \mathbb{F}_2^{kl}$.

1: **for** $i = 1$ to n **do**
2: $\quad k_{i,0} \leftarrow r_i$ ▷ Assign $(k_{1,0}, k_{2,0}, \ldots, k_{n,0}) = (r_1, r_2, \ldots, r_n)$
3: **end for**
4: **for** $j = 1$ to l **do**
5: $\quad k_{n,j} \leftarrow c_j$ ▷ Assign $(k_{n,1}, k_{n,2}, \ldots, k_{n,l}) = (c_1, c_2, \ldots, c_l)$
6: **end for**
7: **for** $j = 1$ to l **do**
8: $\quad$ **for** $i = n$ down to 2 **do**
9: $\quad\quad k_{i-1,j} \leftarrow k_{i,j-1}\backslash k_{i,j}$ ▷ Compute $k_{n-1,j}, k_{n-2,j}, \ldots, k_{1,j}$
10: $\quad$ **end for**
11: $\quad m_j = k_{1,j-1}\backslash k_{1,j}$ ▷ Compute the message block $m_j = k_{1,j-1}\backslash k_{1,j}$
12: $\quad k_{n,j} = k_{1,j} \oplus k_{2,j} \oplus \cdots \oplus k_{n,j}$ ▷ Update the value of $k_{n,j}$
13: **end for**

Output: $M = (m_1, m_2, \ldots, m_l) \in Q^l = \mathbb{F}_2^{kl}$.

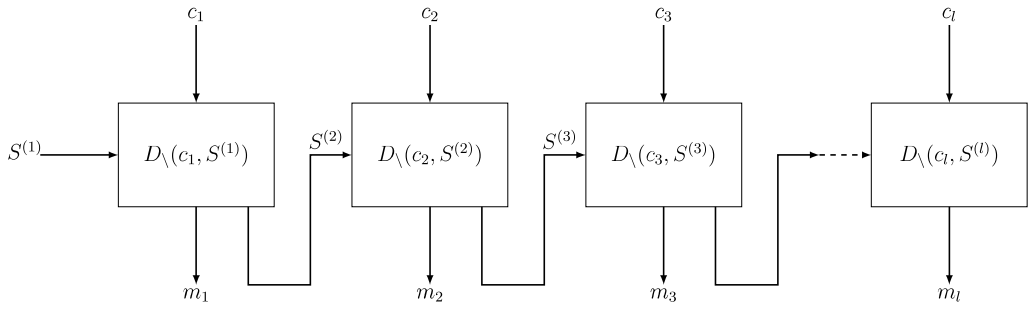

Figure 4.3: Decryption algorithm

Similar to encryption function, here the decryption function, i.e. a function from $Q^l \to Q^l$ can be decomposed into l maps $D_\backslash$ each from $Q \times Q^n \to Q \times Q^n$ and to decrypt the i^{th} block of ciphertext C we compute $D_\backslash(c_i, S^i) = (m_i, S^{i+1})$.

4.3 Security analysis

For the security analysis, we apply the conceptual framework of Bellare et al. [7] for IND-CPA and IND-CCA security assessments.

Theorem 4.3.1. *The symmetric encryption scheme SEBQ as defined in Section 4.2 based on quasigroup is immune to chosen-plaintext attacks. In other words, any adversary having access to encryption oracle, has negligible advantage in distinguishing correct from his chosen two plaintexts, corresponding to any arbitrary ciphertext.*

Proof. In order to prove that the symmetric encryption SEBQ scheme is secure against CPA attack heuristically, first, we consider an experiment (**Exp**) conducted by adversary $\mathcal{A}$ which generates a key, two plaintexts and a ciphertext of randomly selected one plaintext out of those. The experiment outputs 1 if $\mathcal{A}$ is able to identify the correct plaintext. Algorithmically, it is explained below:

Algorithm 7 $\mathbf{Exp}^{cpa}_{\mathcal{A},SEBQ}$(security parameter)

1: The adversary $\mathcal{A}$ is given input 1^* and has access to the oracle $\mathcal{E}_K(\cdot)$. It then produces a pair of messages m_0, m_1 of the equal length.
2: A random bit $b \in \{0, 1\}$ is selected, and subsequently, a ciphertext $c \leftarrow \mathcal{E}_K(m_b)$ is generated and given to adversary $\mathcal{A}$.
3: The adversary $\mathcal{A}$ continues to have the oracle access $\mathcal{E}_K(\cdot)$, and following a series of computations, produces an outputs bit b'.
4: The output of the experiment is 1 if $b' = b$, and 0 otherwise. In the former case, we say that the adversary $\mathcal{A}$ succeeds.

Suppose adversary chooses plaintexts m_0 and m_1 from Q and an initial vector $R \in Q$; and gets a ciphertext C of one of the plaintexts. We consider the cases where adversary can query repeated or non-repeated messages to the oracle.

- **Adversary queries the repeated messages with different initial vectors (IVs)**: Since, the encryption scheme SEBQ is probabilistic (not deterministic) and adversary $\mathcal{A}$ queries the repeated message as many times as it wants. Suppose adversary $\mathcal{A}$ repeatedly queries $m_0 \in Q$ with initial vectors R, with $R' \neq R$ to oracle $\mathcal{E}_K(\cdot)$. Then, the adversary can compute the complete column of Latin square. Eventually, $\mathbf{Exp}^{cpa}_{\mathcal{A},SEBQ} = 1$, i.e. Adversary is able to distinguish the plaintext given its ciphertext with probability 1. Thus, the encryption scheme is insecure. However, this kind of liberty is not given to adversary. So we consider the other case only.

- **Adversary is not allowed to query the repeated messages**: The main idea for adversary to be successful is to construct the Latin square using the adaptive interaction with the $\mathcal{E}_K(\cdot)$ oracle. Adversary queries m_i, $i \notin \{0, 1\}$ to the oracle and get different ciphertexts. This results in detection of whole Latin square except two columns corresponding to m_0 and m_1 as given in Figure 4. However, even with this knowledge adversary is unable to identify whether the given ciphertext corresponds to which message m_0 or m_1 as the R-th row can have C in either column with equal probability, i.e. $Pr[b = 0] = Pr[b = 1] = \frac{1}{2}$, which implies $\mathbf{Adv}_{\mathcal{A}}^{ind-cpa} = 0$.

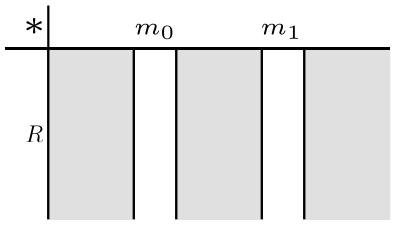

Figure 4.4: Construction of Latin square during IND-CPA attack

□

Proposition 4.3.2. *The SEBQ encryption scheme in not IND-CCA2 secure.*

Proof. Suppose adversary $\mathcal{A}$ has access to both encryption $\mathcal{E}_K(\cdot)$ and decryption $\mathcal{D}_K(\cdot)$ oracle. Upon querying $\mathcal{D}_K(\cdot)$ for $C' \neq C$, the complete Latin square can be computed expect the places where C is present. Using the Definition 4.1.1 of Latin square, those left out places can be easily filled. Thus, $\mathbf{Adv}_{SEBQ}^{ind-cca}(\mathcal{A}) = 1$, hence the proposition follows. □

Similar to the above, it can be readily observed that the same result holds for Left-or-Right indistinguishability. Thus, we aim on applying some transformations to modify the scheme to get indistinguishability under CCA2 attacks.

4.3.1 Unbalanced Feistel transformation

We transform SEBQ scheme using unbalanced Feistel transformation [36] and secure it against adaptive chosen ciphertext attacks. Additionally, using this transformation, the cryptosystems achieves another security goal of non-malleability as introduced by Dolev et al. [42].

Let SEBQ $= \{\mathcal{K}, \mathcal{E}, \mathcal{D}\}$ be a symmetric encryption scheme described in Section 4.2. We proved that it is secure against IND-CPA attack in Theorem 4.3.1 and

subsequently, in Proposition 4.3.2 we have shown that it is not IND-CCA2 secure. So, to shield SEBQ scheme against IND-CCA2 attack we transform the SEBQ into $\widetilde{SEBQ}$ using unbalanced Feistel transformation.

Let M be a variable-length input pseudorandom function, which takes input of any pre-specified length and outputs some fixed length vector. Similarly, let G be a variable-length output pseudorandom function. Our transformation yields a better version of work carried out by [8, 36].

Let us first recall that the fixed-length pseudorandom functions come from a family of keyed multi-set in which functions have fixed (finite) domain and range corresponding to a key. It must be noted that a pseudorandom function is indistinguishable from a random function within the same domain and range. Variable-length input pseudorandom functions accepts an input of arbitrary and variable lengths, generating output of a fixed length. Most of the MACs are examples for this family. The variable-output pseudorandom functions takes fixed length input and outputs a variable length vector of user's choice.

The transformation that we apply to our encryption scheme that makes it secure against CCA depends on security of these variable-length output pseudorandom function.

Let k_0 be the positive integer. The transformation of $SEBQ$ to $\widetilde{SEBQ}$ is describe by following algorithm:

Algorithm 8 Transformed symmetric scheme $\widetilde{SEBQ}$

1: Consider a quasigroup $(Q, *)$ of order 2^k, assuming $Q = \mathbb{F}_2^k$, an initial vector $R = (r_1, r_2, \ldots, r_{k_0}) \in Q^{k_0}$, and message $M = (m_1, m_2, \ldots, m_l) \in Q^l$.

2: Consider a variable-length output pseudorandom function $G : \mathbb{F}_2^{k_0} \to (\mathbb{F}_2^k)^*$.

3: In addition to quasigroup, the key generation algorithm also outputs a positive integer $a > 1$ for variable-length output pseudorandom function G.

4: The transformed scheme modifies the function E_* as described in Subsection 4.2.2 to $\widetilde{E}_*$ as:

$$\widetilde{E}_*\left(m_i, R^{(i)}\right) = E_*\left(m_i, G_a(R^{(i)})\right) \tag{4.5}$$

5: The function $D_\backslash$ as described in Subsection 4.2.3 gets transformed to $\widetilde{D}_\backslash$ as:

$$\widetilde{D}_\backslash\left(c_i, S^{(i)}\right) = D_\backslash\left(c_i, G_a(S^{(i)})\right). \tag{4.6}$$

Recall that the encryption scheme $SEBQ$ is insecure to adaptive chosen-ciphertext attacks as it can be easily observed that adversary can distinguish the correct plaintext from out of two, given ciphertext of any one. This distinguisher

is possible only when message and initial vector R are both elements of Q, but on applying the variable-length output pseudorandom function the length of $G_a(R)$ becomes equal to a, which is a positive integer greater than 1. Now upon querying decryption oracle $\widetilde{D}_{\backslash}(\cdot)$ as described in (4.6), it becomes impossible to recover the quasigroup $(Q, *)$. This is due to the increase in length of initial vector that is to be used in generation of Latin square corresponding to $(Q, *)$. As if initial vector was of length 1, then on querying decryption oracle gives the desired entry of Latin square as explained in Proposition 4.3.2. However, this transformation yields in slight increase of number of bit-operations in the encryption scheme, yet it provides security under adaptive chosen-ciphertext attacks. With these in-sights, we provide the following theorem.

Theorem 4.3.3. *The transformed symmetric encryption scheme $\widetilde{SEBQ}$, based on quasigroup, as described in Algorithm 8 is IND-CCA2 secure.*

4.3.2 Randomness testing

The randomness in the ciphertext plays a crucial role in the security of the symmetric encryption scheme. We analyzed the randomness of the ciphertext generated by Algorithm 5 using the NIST statistical test suite [4]. The results are presented in Tables 4.3, 4.4 and 4.5. We further compared the performance of our scheme against existing symmetric encryption schemes like BCWST [21], INRU [122] in CBC mode and AES-128. For the experiment, we generated a 4000-bit ciphertext sequence using the SEBQ encryption scheme. This is generated using a secret key and a random initial vector, both 400-bits long sequences. The random initial vector was generated using ANSI x9.31 standard random number generation algorithm. The experiments were carried out with a significance level of 99%. Tables 4.3, 4.4 and 4.5 show the success rate and consistency of p-values obtained from the NIST test suite on three different types of plaintexts: 0x00, 0xFF and randomly generated plaintext. These p-values, derived from chi-square tests by partitioning the 0–1 interval into 10 sub-intervals. Our analysis revealed that the randomness of SEBQ encryption is comparable with existing symmetric encryption schemes like AES-128, suggesting its potential security advantages.

4.3.3 Avalanche criterion

When assessing the security of any symmetric encryption scheme, it is crucial to ensure that any modification of input plaintext, random initial vector and key, even a single bit, leads to a significant alteration in the resulting ciphertext [18]. This

Test	SEBQ		BCWST [21]		INRU[122]		AES-128	
	Success %	p value	Sucess %	p value	Success %	p value	Success %	p value
Frequency	100	0.534146	100	0.1025	100	0.1626	99	0.8165
BF	99	0.739918	100	0.6371	100	0.4943	99	0.9914
Runs	100	0.739918	99	0.1626	100	0.6579	98	0.4190
LRO (in a block)	100	0.534146	100	0.5543	98	0.2896	100	0.9716
BMR	100	0.035174	99	0.0457	99	0.1916	98	0.2896
DFT	100	0.350485	98	0.4944	98	0.4372	99	0.7197
OTM	99	0.739918	100	0.5749	99	0.0711	98	0.6993
NOTM (149 templates)	98–100	0.122325–0.991413	96-100	0.0134–0.9978	96–100	0.0066–0.9963	95–100	0.0034–0.9914
MUS	100	0.520354	100	0.475	98	0.1025	100	0.8676
LC	99	0.350485	99	0.5341	99	0.1025	100	0.9114
Serial 1	100	0.534146	99	0.5341	99	0.0965	98	0.2133
Serial 2	99	0.911413	99	0.4011	100	0.5141	99	0.9357
AE	100	0.024011	100	0.0146	100	0.7981	99	0.0220
CSF	100	0.971736	99	0.4373	100	0.0046	99	0.9988
CSB	100	0.965373	100	0.6371	99	0.3345	100	0.6371
RE (8 states)	98.46–100	0.000714–0.524432	98.24–100	0.0008–0.5544	98.46–100	0.0956–0.9411	98.24–100	0.0008–0.5544
REV (18 states)	96.34–100	0.064321–0.974522	98.24–100	0.329–0.9879	95.38–100	0.0704–0.9705	98.24–100	0.0329–0.9879

Table 4.3: A comparative analysis on the success rates and uniformity of the p-value in the NIST Test Suite results for random plaintext across SEBQ, BCWST, INRU and AES-128

Test	SEBQ		BCWST [21]		INRU [122]		AES-128	
	Success %	p value	Sucess %	p value	Success %	p value	Success %	p value
Frequency	99	0.513309	99	0.9717	100	0.4944	97	0.5141
BF	100	0.911413	100	0.6579	99	0.8513	98	0.0135
Runs	98	0.934146	98	0.9781	99	0.6163	97	0.6371
LRO (in a block)	100	0.739918	100	0.6787	99	0.5141	99	0.4559
BMR	100	0.122325	100	0.9357	99	0.5141	97	0.0329
DFT	100	0.534146	100	0.7598	99	0.1537	95	0.6579
OTM	99	0.350485	98	0.5749	98	0.6163	99	0.0167
NOTM (149 templates)	96–100	0.017912–0.991468	96-100	0.004–0.9943	96–100	0.004–0.9978	95–100	0.0102–0.9942
MUS	100	0.52032	98	0.7981	99	0.0668	98	0.5141
LC	99	0.739918	99	0.2133	99	0.6786	99	0.9357
Serial 1	100	0.350485	100	0.1816	100	0.3669	100	0.2897
Serial 2	99	0.534146	99	0.7981	99	0.4559	99	0.1626
AE	100	0.112300	100	0.1154	98	0.3041	99	0.8514
CSF	98	0.876384	98	0.7598	98	0.4190	99	0.2023
CSB	98	0.986294	99	0.5141	99	0.9879	99	0.7399
RE (8 states)	98.46–100	0.200743–0.764423	97.18–100	0.2053–0.8810	98–100	0.0965–0.8832	97.10–100	0.0151–0.7565
REV (18 states)	96.34–100	0.03852–0.974556	95.77–100	0.3863–0.9643	96–100	0.1223–0.9914	94.2–100	0.0151–0.8486

Table 4.4: A comparative analysis on the success rates and uniformity of the p-value in the NIST Test Suite results for plaintext consisting solely of zeros (0x00) across SEBQ, BCWST, INRU and AES-128

Test	SEBQ		BCWST [21]		INRU [122]		AES-128	
	Success %	p value	Sucess %	p value	Success %	p value	Success %	p value
Frequency	100	0.350485	99	0.9558	99	0.6371	100	0.1626
BF	100	0.991468	100	0.3345	100	0.978	99	0.8677
Runs	98	0.350485	100	0.4012	99	0.0519	99	0.4373
LRO (in a block)	100	0.911413	100	0.3346	99	0.2368	100	0.3669
BMR	99	0.066882	99	0.1719	99	0.0519	100	0.5955
DFT	98	0.035174	97	0.9643	98	0.3505	95	0.6163
MUS	99	0.499323	99	0.5141	99	0.1223	100	0.4943
OTM	99	0.122325	98	0.6163	99	0.9114	99	0.1626
NOTM (148 template)	96–100	0.008879–0.911413	96-100	0.0046–0.9915	96–100	0.0076–0.9914	95–100	0.0179–0.9781
LC	98	0.066882	98	0.6993	100	0.1537	100	0.9463
Serial 1	99	0.534146	98	0.0205	99	0.3505	100	0.3505
Serial 2	99	0.739918	98	0.1373	99	0.2757	100	0.2757
AE	100	0.112300	100	0.6993	100	0.4749	95	0.3345
CSF	98	0.994708	99	0.8832	98	0.4349	100	0.7399
CSB	98	0.823133	99	0.6579	100	0.9558	99	0.6993
RE (8 states)	98.46–100	0.120073–0.876423	96.88–100	0.1480–0.8623	98.21–100	0.00003–0.9114	98.55–100	0.0514–0.8755
REV (18 states)	96.34–100	0.08522–0.975564	98.44–100	0.0822–0.9114	94.64–100	0.0020–0.9558	97.1–100	0.0190–0.7887

Table 4.5: A comparative analysis on the success rates and uniformity of the p-value in the NIST Test Suite results for plaintext consisting solely of ones (0xFF) across SEBQ, BCWST, INRU and AES-128

property is also known as diffusion in ciphertext or the avalanche effect. The concept of avalanche criteria has been mathematically defined in various ways across different contexts, including block ciphers, Boolean functions and hash functions. We analyze the avalanche criteria of the SEBQ scheme. Simultaneously compared the results with the existing schemes like INRU [122], BCWST [21] and AES-128.

Avalanche analysis by modification in key

We analyze ciphertext changes for the SEBQ scheme, whenever a key alteration occurs, specifically when the Latin square is modified. To assess whether SEBQ fulfills the avalanche criterion, we conducted experiments with 10 randomly generated Latin squares. We encrypted a randomly generated plaintext comprising 4000 bits long sequence using each unique secret keys along with the 400-bit long initial vector and Algorithm 5. The outcomes of the experiments are mentioned in the Table 4.6. A comparative analysis of the avalanche effect in SEBQ with INRU [122], BCWST [21] and AES-128 is also mentioned in Table 4.6.

Encryption scheme	Max % change in ciphertext	Min % change in ciphertext
BCWST [21]	50.054	49.931
INRU [122]	50.164	49.822
AES-128	50.046	49.937
SEBQ	**50.352**	**49.90**

Table 4.6: Comparative analysis of avalanche effect of secret key

Avalanche analysis in ciphertext by changing the random initial vector

In this section, we examine how a single bit flip in the initial vector affects the ciphertext of SEBQ scheme. We use a randomly generated 16×16 size Latin square as the secret key to encrypt a randomly generated 4000-bit plaintext. To assess the avalanche effect of initial vector on the ciphertext, we generate 100 random initial vectors and encrypt the same plaintext using each of them. We analyzed the avalanche effect caused by flipping each bit position within the initial vectors. The results are shown in Table 4.7. The final rows of Table 4.7 present the maximum and minimum avalanche effects of the initial vector on ciphertext. This experimentation is repeated 10 times and the outcomes of the avalanche effect being stored in the Table 4.7.

Pos.	Percentage change in ciphertext with corresponding experiment										
	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}	Avg.
1	50.600	50.175	48.800	50.775	50.249	49.350	49.650	48.500	50.349	50.200	49.865
2	49.475	51.500	49.225	50.700	51.249	48.699	48.650	49.850	50.100	50.649	50.010
3	50.275	50.050	49.850	49.700	50.100	49.550	50.949	49.350	50.824	50.550	50.120
4	50.349	50.625	50.800	49.875	51.500	49.650	51.375	49.100	50.700	49.575	50.355
5	50.800	50.449	49.825	49.850	49.325	50.475	50.125	50.900	50.975	49.875	50.260
6	50.724	50.075	49.500	50.149	50.749	48.675	50.025	50.375	50.449	50.550	50.127
7	50.400	49.200	49.875	49.675	**48.350**	50.525	51.425	49.350	50.400	50.575	49.977
8	49.400	50.900	50.824	49.850	49.925	50.025	49.875	48.750	50.525	49.800	49.987
9	50.875	50.600	49.050	50.400	50.824	48.375	50.175	49.425	50.375	51.775	50.187
10	49.500	50.775	50.375	49.450	50.075	49.800	49.875	50.224	51.200	50.224	50.150
128	50.600	50.525	50.600	50.849	50.324	51.475	50.600	49.825	50.175	49.550	50.452
256	49.900	50.449	51.175	50.625	50.449	48.900	**51.824**	49.900	49.250	49.875	50.235
Maximum % change in ciphertext is 51.824											
Minimum % change in ciphertext is 48.350											

Table 4.7: Observation of avalanche effect due to change in initial vector bit positions

Avalanche analysis in ciphertext by change in plaintext

To examine how modifying a single bit in the plaintext affects the resulting ciphertext of SEBQ, we conducted a series of experiments. We generated 100 random plaintext, each consisting of a 4000-bit long vector. We then individually encrypted each plaintext using a randomly generated secret Latin square of order 16×16 in conjunction with a 400-bit long initial vector. Subsequently, we compared the avalanche effect caused by a single plaintext bit flip in SEBQ scheme with the existing schemes like INRU [122], BCWST [21] and AES-128. The results are presented in Table 4.8. To ensure statistical significance, we repeated these experiments 10 times. The avalanche effect values for all repetitions are shown in Table 4.9.

Encryption scheme	Max % change in ciphertext	Min % change in ciphertext
BCWST [21]	50.041	49.959
INRU [122]	50.042	49.943
AES-128	50.032	49.957
SEBQ	**52.550**	**48.000**

Table 4.8: Comparative analysis for maximum and minimum avalanche effect of plaintext

Pos.	Percentage change in ciphertext with corresponding experiment										
	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}	Avg.
1	49.875	51.200	52.050	50.050	49.650	48.525	49.925	49.300	50.625	50.800	50.200
2	49.875	50.375	49.075	50.949	51.075	50.625	48.949	49.450	50.349	50.324	50.105
3	49.375	49.750	50.050	50.675	49.825	50.800	51.025	50.849	49.475	48.850	50.067
4	49.400	51.675	49.675	50.625	49.000	51.050	49.225	50.050	49.425	50.125	50.025
5	49.150	50.324	49.550	50.324	50.575	50.550	50.349	49.675	50.275	48.699	49.947
6	49.600	49.825	50.400	**48.000**	50.125	50.949	51.100	51.200	49.375	51.050	50.162
7	49.675	49.950	49.900	49.075	49.675	**52.550**	49.950	49.700	49.625	49.375	49.847
8	50.324	49.525	51.200	49.025	49.700	50.149	50.075	49.775	49.950	49.920	49.965
9	50.025	50.000	49.925	50.224	49.525	50.975	47.500	50.300	51.200	50.525	50.020
10	49.425	48.725	50.775	49.250	49.500	51.100	50.324	49.775	50.600	50.675	50.015

Table 4.9: Observation of avalanche effect due to change in plaintext bit positions

4.4 Analysis of the scheme

Consider $Q = \mathbb{F}_2^k$ and a Latin square of order 2^k. Now, let's assume that message $M = m_1, m_2, \ldots, m_l \in (\mathbb{F}_2^k)^l$ is being encrypted with the given Latin square in conjunction with the initial vector $IV = r_1, r_2, \ldots, r_n$.

Proposition 4.4.1. *The number of operations required to encrypt the given message M using the given Latin square and the initial vector IV is $n+(l-1)(n+k)$.*

Proof. The number of operations needed to encrypt the i^{th} block of message M is $(n+k)$ and for the entire message M is $(l-1)(n+k)$ operations. Additionally, updating the vector requires performing n xor operations. Therefore, the total number of operations required for the encryption scheme is equal to $n + (l-1)(n+k)$. □

Proposition 4.4.2. *The number of operations required to decrypt the given ciphertext C using the given Latin square and the initial vector IV is $n+(l-1)(n+k)$.*

Proof. Proof is similar as above Proposition 4.4.1. □

Hence, the total operations required for the symmetric encryption scheme SEBQ are equal to $2(n + (l-1)(n+k))$ which is proportional to $\mathcal{O}(nl)$.

Consider a scenarios where a user intends to secure a message $M \in \mathbb{F}_2^{64}$ through encryption and by using 128-bit long secret key along with 16-bit long initial vector. The number of operations required to encrypt the message M is just 70. In contrast, for the same message and parameter set, the INRU: A quasigroup based lightweight block cipher [122] necessitates 154 operations and to encrypt the same message with same parameter 120 number of operations are needed to perform in the encryption scheme BCWST [21]. In a nutshell, the symmetric encryption

scheme SEBQ described in Algorithm 5 demonstrated superior efficiency compared to the lightweight block ciphers proposed in [122], [21].

Suppose $Q = \mathbb{F}_2^4$ and a Latin square of order m has been utilized to encrypt the message $M = m_1, m_2, \ldots, m_l \in \mathbb{F}_2^{4l}$ where $m_i \in \mathbb{F}_2^4$ in conjunction with random vector $R = r_1, r_2, \ldots, r_8 \in \mathbb{F}_2^{32}$ where each $r_i \in \mathbb{F}_2^4$. Suppose adversary $\mathcal{A}$ possesses information about the 128-bit ciphertext. The number of operations needed to make an educated guess about the plaintext corresponding to the provided ciphertext are detailed in Proposition 4.4.3 and 4.4.4. The minimum order of Latin square to provide security against 128-bit and 256-bit known ciphertext attack is given by following Proposition 4.4.3 and 4.4.4.

Proposition 4.4.3. *In order to attain 128-bit security against known ciphertext attack in SEBQ, it is imperative that the order of Latin square must be greater than 11.*

Proof. According to the Proposition 4.4.2 the decryption of a 128-bit ciphertext necessitates 380 operations. In order for an adversary $\mathcal{A}$ to successfully guess the correct plaintext corresponding to given ciphertext it needed to perform operations greater than $L(m) \times 380$ where $L(m)$ represents the count of Latin squares of order m. To achieve 128-bits security level, parameter m ought to be selected in a manner that $L(m) \times 380 \geq 2^{128}$ which implies that the parameter m should be greater than 11. Conclusively, achieving a 128-bit security level implies that order of Latin square should be greater than 11. □

Proposition 4.4.4. *In order to attain 256-bit security against known ciphertext attack in SEBQ, it is imperative that the order of Latin square should be greater than 13.*

Proof. Proof is same as Proposition 4.4.3. □