


## Research Article

## Development of Robust and Efficient Symmetric Random Keys Model based on the Latin Square Matrix

Nada Hussein M. Ali <sup>1</sup>, , Mays M. Hoobi <sup>1,\*</sup>, , Dunia F. Saffo <sup>1</sup>, <sup>1</sup> Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq.

## ARTICLE INFO

## Article history

Received 17 Aug 2024

Accepted 16 Nov 2024

Published 10 Dec 2024

## Keywords

Brute force attack

Key Generation

Latin Square Matrix

PRNG

Chosen Ciphertext



## ABSTRACT

Symmetric cryptography forms the backbone of secure data communication and storage by relying on the strength and randomness of cryptographic keys. This increases complexity, enhances cryptographic systems' overall robustness, and is immune to various attacks. The present work proposes a hybrid model based on the Latin square matrix (LSM) and subtractive random number generator (SRNG) algorithms for producing random keys. The hybrid model enhances the security of the cipher key against different attacks and increases the degree of diffusion. Different key lengths can also be generated based on the algorithm without compromising security. It comprises two phases. The first phase generates a seed value that depends on producing a randomly predefined set of key numbers of size  $n$  via the Donald E. Knuths SRNG algorithm (subtractive method). The second phase uses the output key (or seed value) from the previous phase as input to the Latin square matrix (LSM) to formulate a new key randomly. To increase the complexity of the generated key, another new random key of the same length that fulfills Shannon's principle of confusion and diffusion properties is XORed. Four test keys for each 128, 192, 256, 512, and 1024-bit length are used to evaluate the strength of the proposed model. The experimental results and security analyses revealed that all test keys met the statistical National Institute of Standards (NIST) standards and had high values for entropy values exceeding 0.98. The key length of the proposed model for  $n$  bits is  $25*n$ , which is large enough to overcome brute-force attacks. Moreover, the generated keys are very sensitive to initial values, which increases the complexity against different attacks.

## 1. INTRODUCTION

Cryptography protects confidentiality by altering the source data (plaintext) into an unreadable form (ciphertext). Additionally, it is referred to as secret writing to prevent stealing confidential data, modifying and deleting it, and many other problems [1, 2]. This method guarantees that only the intended recipient can recognize the actual content of the message, thereby preventing unauthorized access [3]. Hence, the encryption method must have a strong and complicated key that unauthorized users cannot easily guess or crack [4, 5]. Moreover, generating cipher keys randomly is considered an essential principle in cryptography and information security science to protect confidentiality, integrity, and availability [6, 7]. Many cipher algorithms depend particularly on pseudorandom number generator (PRNG) sequences for cipher keys, which are frequently employed in cryptography and digital communications [8]. The key generation process must therefore strike a balance between randomness, unpredictability, and efficiency to ensure security without introducing unnecessary vulnerabilities. Cryptanalysis, on the other hand, is a method of breaking cipher text created by a cryptographic algorithm [9]; thus, the randomness of cryptographic keys has a major effect on the system's security. Therefore, achieving high levels of key randomness is crucial for any cryptosystem enhancement, implementation, and development [10].

The Latin square matrix (LSM) contains exactly  $M$  distinct elements in  $M \times M$  of the matrix. When each element appears only once per row and column,  $A$  is referred to as an LSM, as illustrated in Figure 1, with various orders. The most effective method for constructing an LSM is to start with its standard form, which is achieved by shifting the integer elements of an array from 1 to  $M$ . A new LSM can be created by rearranging the rows or columns of this standard form [11, 12]. Generally, constructing an LSM via the offline algorithm takes  $O(n^2)$ , which is optimal for random key generation [13][27].

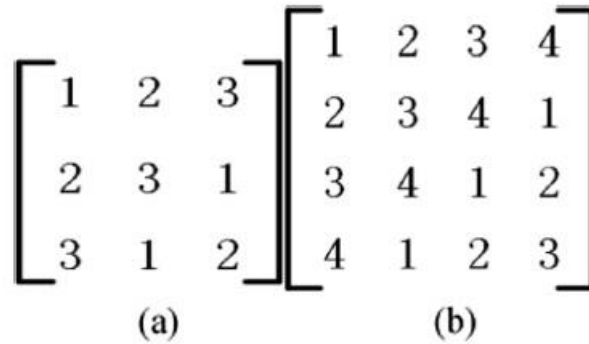


Fig. 1. (a) Order Three of Latin Square, (b) Order Four of Latin Square [11].

The aim of this research is to combine the mathematical structures of Latin square matrices (LSMs) and the PRNG to produce a robust cryptographic key. The idea behind using LSMs for cipher key generation is to exploit their inherent properties, such as their uniform distribution of symbols and their resistance to regular patterns, to produce keys that are both robust against cryptanalysis and difficult to predict or replicate.

The structure of the remaining sections of this paper is divided as follows: Section 2 presents the state-of-the-art concerning related work, while Sections 3 and 4 present the proposed hybrid model and the analysis of the results, respectively. Finally, Section 5 presents the conclusion.

## 2. RELATED WORKS

Various studies suggest different techniques to solve the weakness in key generation for the cipher process. [6] suggested a developed pseudo-random sequence generator (PRSG) to solve the problem of the nonuniform distribution of the sequences. It is based on nonintegral numbers with a modular number operation system, and the statistical empirical tests for PRSGs have passed and approved its success. The sort-index method and diffusion processing for image encryption are performed depending on the PRSG. The obtained results showed good statistical characteristics and increased the degree of security in cryptographic applications. [14] suggested a key generation method based on 2D Henon chaotic map improvement. This process is implemented among mantissa bits in a consecutive sequence via a simple XOR operation. Pseudorandom bit generators (PRBGs) are tested based on the National Institute of Standards (NIST) statistical performance and Hamming distance metrics. Furthermore, various security analyses, such as brute force, differential attack, and other tests, have been performed. The ease of this approach makes it compatible with a variety of software and hardware platforms[26].

Moreover, [15] proposed a system for key generation involving a combination of a stream cipher and evolutionary algorithms. It is composed of three phases: a logistic map, a left-feed shift register (LFSR), and a genetic algorithm. Combining LFSR and the logistic map stream yields a chaotic image that is used to extract random values and is subsequently employed by the genetic algorithm to produce random keys. The results evaluated by NIST statistics revealed that the produced sequences are highly random. In addition, brute force, differential, and guess-and-determine attacks are also used, which makes them suitable for cryptographic applications. However, [16] uses reinforcement learning techniques to generate dynamic random numbers by evaluating and selecting all possible states of an episode, ensuring the randomness of the numbers. It uses the long short-term memory (LSTM) method to retain long-term memory (LTM) of past patterns and to guide the selection of new patterns based on these previous observations. The results assessment confirms the randomness and security of the generated numbers. Additionally, [17] proposed a hybrid method for generating pseudorandom numbers by combining LFSR and linear congruent generators (LCGs). The results were evaluated based on randomness, correlation between the keys, and the impact of changing the initial state on the generated keys. Additionally, the obtained results passed many tests, such as brute-force and differential attacks.

A novel adaptive image encryption system was proposed in [18] based on the LSM and the fractional-order Lu system. The suggested method of image encryption combines the LSM and chaotic sequences. The results show that this approach is effective for resisting known plain image attacks (KPIAs) and chosen plain image attacks (CPIAs). An innovative two-stage algorithm for generating orthogonal Latin squares within finite fields for bolstering the efficiency of image encryption was presented in [19][28]. It exploits the property of the orthogonal LSM to generate randomized ones that correlate with

the plaintext. Security testing of the cipher image revealed that the number of changing pixels rate (NPCR) and unified averaged changed intensity (UACI) closely align with the expected values of 99.6094% and 33.4635%, respectively.

The primary contribution of this work is the introduction of a novel hybrid model that combines the LSM and SRNG algorithms to create secure random keys across multiple layers. This model produces cipher keys of the desired length (128, 192, 256, 512, and 1024-bit) depending on the symmetric algorithm applied. This approach significantly enhances complexity and unpredictability in key generation by expanding the key space and increasing sensitivity to initial parameters. As a result, the model strengthens the security of cryptographic systems, offering improved resistance against attacks that exploit predictable key patterns.

### 3. MATERIALS AND METHODS

The primary objective of this section is to introduce the proposed model for random key generation based on the LSM approach. The components of the proposed model can be recapitulated into two phases:

- **The first phase:** Initially, this phase produces a randomly predefined set of key numbers each of size  $n$  via Donald E. This phase involves the subtraction random number generator algorithm (subtractive method). Depending on the cipher key length needed, it can select any number of initial keys within the set and merge them to formulate a final cipher key. The outcome of this phase is fed to the next phase, as described in Algorithm 1.

<b>Algorithm (1): Phase1 of Random Key Generation</b>
<b>Input:</b> Max : represents maximum number generated in the set
<b>Output:</b> B-Key (binary Key of $n$ -bits).
<b>Begin</b> <b>Step 1:</b> Repeat (For $i= 1 \dots \dots \text{Max}$ ) <b>Step 2:</b> Generate $R[i]$ decimal random numbers each have a size of 4 bytes using the SRNG method and save it in $R[i]$ <b>Step 3 :</b> End For <b>Step 4:</b> Select randomly $M$ numbers from $R[i]$ . // $M$ : is a number of digits required to generate a key length of $n$ -bits. <b>Step 5:</b> Concatenate the $M$ digits to form a final D-Key (in decimal). <b>Step 6:</b> Convert decimal D-Key to a binary B-Key of $n$ -bits length. <b>End</b>

- **The second phase:** This phase was implemented to increase the randomness of the key generated from the previous phase based on the LSM. Notably, the 2D LSM in the proposed model depends on the number of bits in the key, which is generated from phase one. For example, if the key length is 128 bits, then the LSM is composed of 128 rows and 128 columns. Algorithm 2 clarifies in detail the steps of the final key generated from this phase. The purpose of step 6 in Algorithm 2 is to generate another key ( $R$ -Key) to increase the degree of complexity of the key generated.

<b>Algorithm (2): phase2 of Random Key Generation</b>
<b>Input:</b> B-Key (binary key of $n$ -bits output from Algorithm1)
<b>Output:</b> Final-Key (binary key of $n$ -bits)
<b>Begin</b> <b>Step 1:</b> Generate Latin square matrix $T [n,n]$ . <b>Step 2:</b> Fill every row in $T [n,n]$ of value $0, \dots, n-1$ randomly. <b>Step 3:</b> Select three random row indices in the $T [n,n]$ , $RW1, RW2, RW3$ . <b>Step 4:</b> Permuted B-Key bits according to the sequence of $RW1$ contents $\rightarrow$ per-key1. <b>Step 5:</b> Permuted per-key1 bits according to the sequence of $RW2$ contents $\rightarrow$ per-key2. <b>Step 6:</b> Permuted per-key2 bits according to the sequence of $RW3$ contents $\rightarrow$ per-key3. <b>Step 6:</b> Generate randomly a new $R$ - Key of $n$ -bits by SRNG algorithm. <b>Step 7:</b> Apply the following formula to obtain the (Final-Key) $B$ - Key: $(\text{Final-Key}) B - Key = \text{per} - \text{key3} \oplus R - Key$ <b>End</b>

Figure 2 depicts in detail the main structure of the two phases, and Figure 3 graphically shows an example of generating a 128-bit random key suggested by the proposed model yields after two-phase implementation.

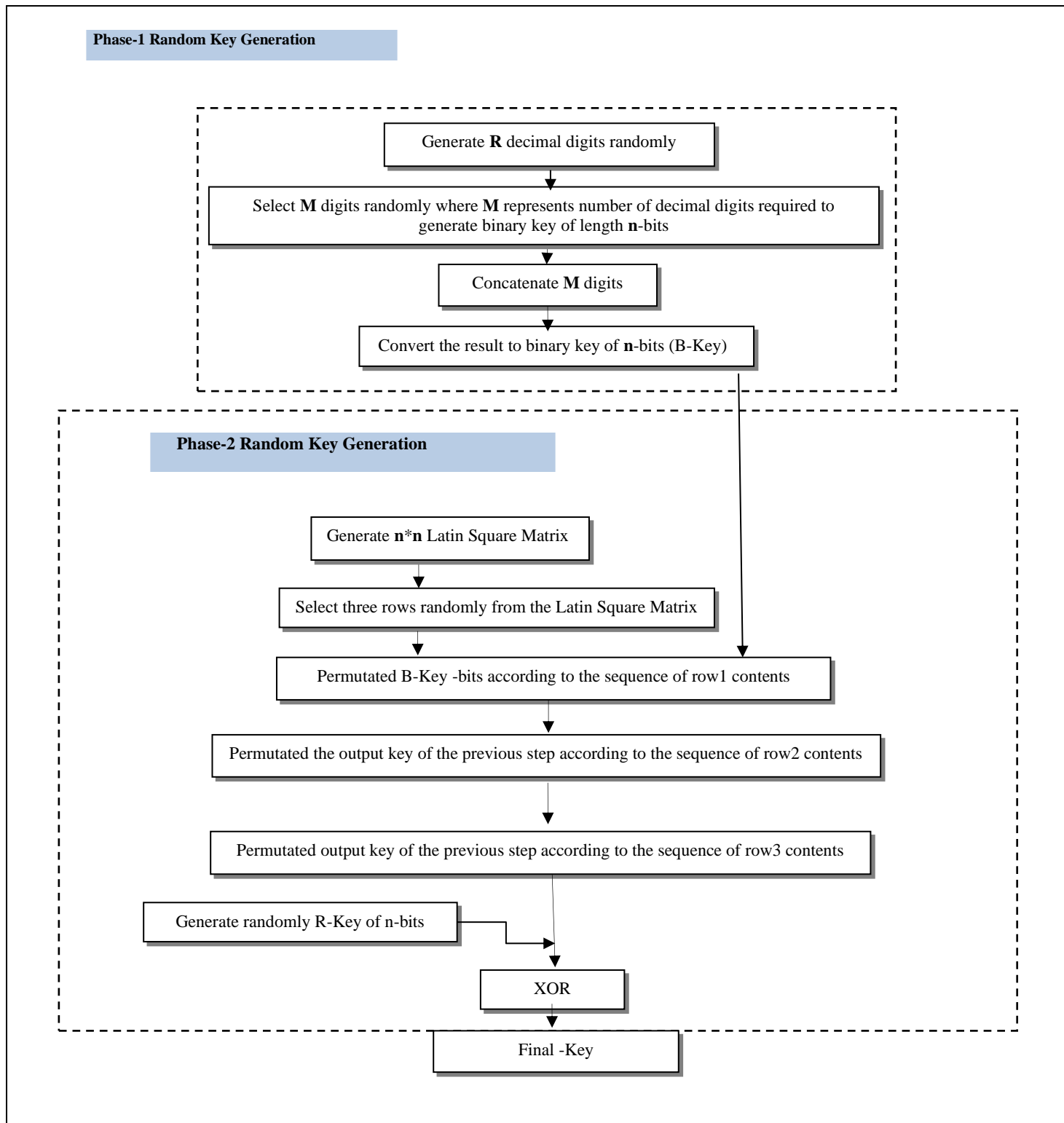


Fig. 2. A Block Diagram for the Proposed Model

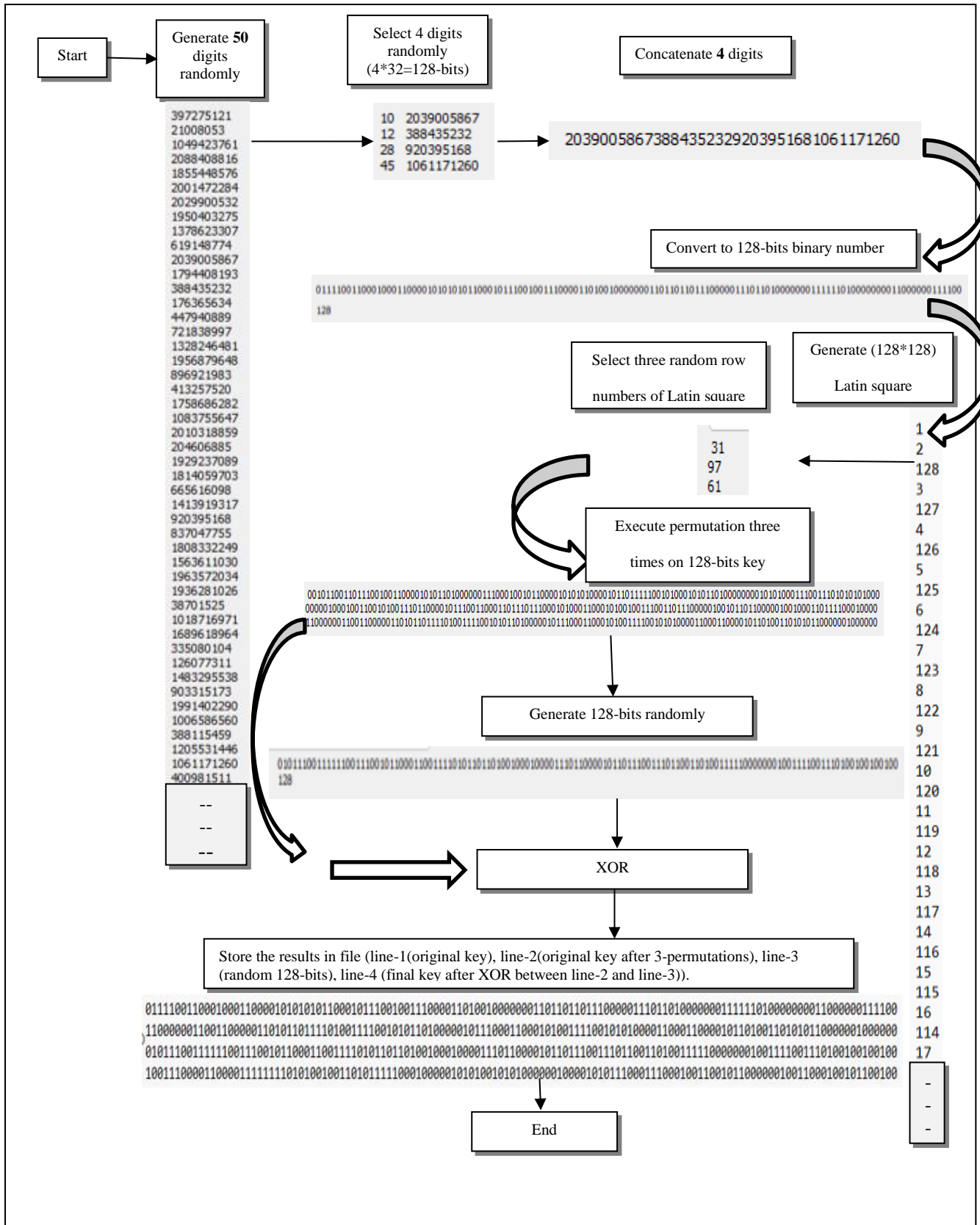


Fig. 3. Example of 128-bit key generation implemented with two phases

Figure 2 shows the general structure of the proposed two-phase model. Its steps can be summarized as follows:

- Step 1:** Generate R decimal digits via Donald E. In the SRNG, each digit has 4 bytes or 32 bits.
- Step 2:** Select the number of digits randomly to formulate the cipher key. For example, if the desired key length is 128 bits, then select 4 digits randomly, concatenate them, and then convert them to binary form to obtain the final key of length n-bits.
- Step 3:** Depending on the n bits obtained from step 2, a 2D LSM is generated.
- Step 4:** Fill each row randomly in 2D LSM with numbers (0,.....n-1) without repetition.
- Step 5:** Randomly select three-row indices from the 2D LSM.
- Step 6:** Depending on the three rows of content, the final key obtained from Step 2 is consequently scrambled three times.
- Step 7:** As an extra step, the SRNG generates a random key of size n and XORed with the key obtained from Step 5. to increase security.

#### 4. RESULTS AND DISCUSSION

The experimental tests were run on a Windows 11 Home processor with an Intel(R) Core (TM) i3-1005G1 CPU @ 1.20 GHz and 1.20 GHz. Random Access Memory (RAM): 8 GB. The system type is a 64-bit operating system with an x64-based processor. For the experimental work, Table 1 represents the four test keys of lengths 128, 192, 256, 512, and 1024, which are considered to test the quality of randomness obtained from the key generation phases of the proposed model. To assess the effectiveness of the proposed model, several measurements and attacks were conducted on the generated keys to prove their efficiency, strength, and degree of complexity as follows:

TABLE I. FOUR TEST KEYS OF VARIOUS LENGTH IN HEXADECIMAL

Key no.	Key Value/Hex	Key no	Key Value/Hex
<b>128-bit</b>			
1	2B23E4BF13955F5FD54171BFDBCBD40	3	B2D2B590B40567B23809364BC94349DE
2	966404998E83DE04BAF3369A6A7A876A	4	DB9F46D498D376C2F0C7D5D51F5FC2CB
<b>192-bit</b>			
1	DEA913EECA49E5F0965FF859AC0F4D9 C0D7EB9D0FAA89DC9	3	B4D8D371FC9BA7B1BB30AE86732EA34183269675 0681A18C
2	9D3E0234451332BE736B5FC90E1B5347B A2EB139EF4187AD	4	D79089EFC9D606AA0D602CB49A62609F5D3E1CB AE8FC4224
<b>256-bit</b>			
1	4A6FE1922586CBD42E93E420F3EB3EF9 6785E9D3E3F277463F3E392EDAAB6C5A	3	3AA128085F91BAF025F6A8865EEDB2AD4305C978 EF599F16B47BB19D4CA98E0A
2	96226C01979C0F9F6DA233664CB13CC98 58F3DC779DDE9B65ED9AE89D0507133	4	4AD0F33D0EA318E3DF59E7B5E65DB0EC1110EAB D0245D073466710614F9FA497
<b>512-bit</b>			
1	3D4497FB895A333EDC270631A87C81396 E7C504248240EC0AC6E01D394BCCABA 729BCA4D713C253214AA390099E40C14 E634D5429E2A034BC5B14FAC6510D4E0	3	77C7499446CAEE1C86AFE87CDDAD80A6116102B E36EA53331175A7344DC65F7974DE76A71A72EFC DA7BF7ECB482170895A6BA98609316147219B5081 A83F92FE
2	F523491D1785DDB6FA6C3C44E5925514 BB94C0EAFD712F3B384B68EBBB101BB EECB7FE6513C4BA7292D3037C8293E5E 21DE84DFAD890A530CE4BC6820E48612 3	4	553B5969C76D243554A8CEF43071B3328109F9A75 0ECECD0A8CF5E291089127FE139F48C75F8295C97 832BF6CB2D5A7260391C34E00232701D72DF4AA3 08DEDF
<b>1024-bit</b>			
1	7BAC357E25C94D9094D16F291BB2FD81 9E6E2FA2D1FD91AD20CD4F2B54F1D93 33EB931443EB68109F5B1196FABBF3CC 639056056C2F76D501040D09002B9749C B3B64E3696D5B3757F1DDDD862697C1C9 8FA886C71006C607356ABF376DC332039 2E0C804C444C5E146C85C8E375707185E 3B4A6BFF734BF6206DD2A32C4FE8C5	3	A5968EB32A3EC739C44EF5D45E033FA853331BB0 230B51B0EC5F40792ABA2A5F28C06C2A18EB0F2 A159819240EE8AD22FD7E5F04BB3954E35534D095 979CE688FC6D59593844A2DC387D9615A588063A 19212833C12477C8FD96F689C17CB874065411741D F04554BE5C76D66679106B183501AFA6FF3809AF4 66C7B933CE12

2	F53858141815334714AA27E7B154BB0506 2543C6D0317C9548CE2805D0987E454C2 DA0557325C46AAA504B42104A57B6CC 7992231712E86BE4B0EEC6541145B39AD 3BD2D136C16F047BC0D6DBD09F7F751 C51982EF8FA176C80A6D71857C7136AB 236449C1A408D72ED2D6D2051D532E28 F1FF7B9A90D6854A49D780BB80013A	4	6FBF351A908517C6DA86655BFA4C080CDC03EBA 2AA691C5F5192686C950224CBB6C6D4A0CBF85C CB4E2686EBE1EE92566392810DA984986DBDAFD A28485C70E6097246FBA9D58836D2219609211CA9 F8919FE49D14A2594FACF3F3E19CD7CE0920986B 37FA846C280368DCD8C2FF1F62D5EC1510CC92EE 0D13B7CCD1AD2723E8
---	--	---	--

### 4.1 NIST Performance Evaluation Tests

The National Institute of Standards and Technology (NIST) test suite is an arithmetic package that includes numerous tests designed to examine the unpredictability of (randomly lengthy) binary series generated by either hardware- or software-based cryptographic PRNGs. These assessments stress a variety of different types of nonrandomness that might exist in a series [20]. Statistical tests were applied to measure the strength of the keys. The key lengths used are 128 bits, 192 bits, 256 bits, 512 bits, and 1024 bits. As shown earlier in Table 1, four different values for each key length were used to test the NIST effectiveness and accuracy performance of the proposed model, as shown in Tables 2, 3, 4, 5, and 6. Notably, only six NIST tests are used because of the limited length of the generated keys for p values  $\geq 0.01$ .

TABLE II. NIST TESTS FOR FOUR 128-BIT KEY LENGTHS

Test type	Key1	Key2	Key3	Key4	Decision
Frequency (Monobit)	0.077099	0.723673	0.376759	0.111611	Pass
Frequency within a Block	0.077099	0.723673	0.376759	0.111611	pass
Runs	0.918815	0.472228	0.1356006	0.441729	pass
Longest Run of Ones in a Block	0.539089	0.682747	0.276999	0.059137	pass
Approximate Entropy Test	0.034025	0.840568	0.486691	0.473553	pass
Cumulative Sums (Cusum) Test	<b>P value Forward</b>				
	0.0431129	0.574763	0.369655	0.154199	pass
	<b>P value Reverse</b>				
	0.067789	0.946011	0.499938	1	pass
Success Ratio	6/6	6/6	6/6	6/6	

TABLE III. NIST TESTS FOR FOUR 192-BITS KEY LENGTHS

Test type	Key1	Key2	Key3	Key4	Decision
Frequency (Monobit)	0.193930	0.665005	0.665005	0.563702	Pass
Frequency within a Block	0.288844	1	0.479500	0.479500	pass
Runs	0.788328	0.874438	0.762269	0.980774	pass
Longest Run of Ones in a Block	0.503840	0.559409	0.111428	0.965326	pass
Approximate Entropy Test	0.467892	0.968090	0.068824	0.881931	pass
Cumulative Sums (Cusum) Test	<b>P value Forward</b>				
	0.297799	0.880822	0.619770	0.686526	pass
	<b>P value Reverse</b>				
	0.693833	0.747681	0.297799	0.777852	pass
Success Ratio	6/6	6/6	6/6	6/6	

TABLE IV. NIST TESTS FOR FOUR 256-BITS KEY LENGTHS

Test type	Key1	Key2	Key3	Key4	Decision
Frequency (Monobit)	0.077099	0.723673	0.376759	0.215924	Pass
Frequency within a Block	0.077099	0.723673	0.376759	0.215924	pass
Runs	0.918815	0.472228	0.1356006	1.437248	pass
Longest Run of Ones in a Block	0.539089	0.682747	0.276999	0.201226	pass
Approximate Entropy Test	0.034025	0.840568	0.486691	0.505500	pass
Cumulative Sums (Cusum) Test	<b>P value Forward</b>				
	0.0431129	0.574763	0.369655	0.223219	pass
	<b>P value Reverse</b>				
	0.067789	0.946011	0.499938	1	pass
Success Ratio	6/6	6/6	6/6	6/6	

TABLE V. NIST TESTS FOR FOUR 512-BITS KEY LENGTHS

Test type	Key1	Key2	Key3	Key4	Decision
Frequency (Monobit)	0.929568	0.723673	0.0170102	0.536101	pass
Frequency within a Block	0.492275	0.487445	0.113095	0.707046	pass
Runs	0.658276	0.654459	0.601197	0.583927	pas
Longest Run of Ones in a Block	0.269774	0.843861	0.549406	0.397314	pass
Approximate Entropy Test	0.514679	0.942272	0.090820	0.899543	pass
Cumulative Sums (Cusum) Test	<b>P value Forward</b>				
	0.536609	0.314554	0.034020	0.536609	pass
	<b>P value Reverse</b>				
	0.807114	0.574763	0.504677	0.737518	pass
Success Ratio	6/6	6/6	6/6	6/6	

TABLE VI. NIST TESTS FOR FOUR 1024-BITS KEY LENGTHS

Test type	Key1	Key2	Key3	Key4	Decision
Frequency (Monobit)	0.802587	0.211299	0.028706	0.317310	Pass
Frequency within a Block	0.041494	0.421356	0.139089	0.764234	pass
Runs	0.898971	0.764722	0.074607	0.552291	pass
Longest Run of Ones in a Block	0.388893	0.542961	0.004957	0.163534	pass
Approximate Entropy Test	0.571045	0.791973	0.034536	0.518181	pass
Cumulative Sums (Cusum) Test	<b>P value Forward</b>				
	0.857964	0.221980	0.048897	0.469326	pass
	<b>P value Reverse</b>				
	0.814611	0.625703	0.038181	0.689269	pass
Success Ratio	6/6	6/6	6/6	6/6	

The influence of the key length has a major effect on each NIST test since it increases the randomness of the key values. If the P value is  $\geq 0.01$ ; then the key values have a uniform distribution; which proves the success of the proposed model. Each test shows the success of breaking down the coefficient correlation between key values based on the seed value by the proposed model.

**4.2 Information Entropy Analysis**

Information entropy, considered a key parameter in information theory, describes the degree of chaos in a system. This method involves gathering randomness using cryptography or other applications that call for random data from an operating system or application [21]. The encryption data should be sufficiently muddled to withstand a statistical analysis attack. In this case, information entropy may be utilized in the calculation, as the more complicated the cipher data are, the higher the first entropy is. Equation (1) is used to compute first-order information entropy [22, 23].

$$H(m) = \sum_{i=0}^{2^n-1} P(m_i) \log_2 \frac{1}{P(m_i)} \tag{1}$$

where the number of bits is denoted by the variable  $n$ , and the total number of symbols is denoted by the variable  $M (= 2^n)$ . The variable  $mi \in M$  and the variable  $P(m_i)$  denote the probability of having  $mi$  levels in the key space.

Entropy measures the unpredictability of a cryptographic key and is frequently used in cryptanalysis. To break a key via a brute force attack, one typically needs to try, on average,  $2^{n-1}$  attempts, where  $n$  represents the number of bits in the key. Based on the obtained test results, as shown in Table 7, the suggested approach creates keys with a sophisticated entropy value.

TABLE VII. RANDOM KEYS ENTROPY TEST

Key length	Key1	Key2	Key3	Key4
128-bit	0.982317	0.999295	0.995593	0.985688
192-bit	0.993651	0.999295	0.999295	0.998747
256-bit	0.992546	0.999604	1	1
512-bit	0.999989	0.999824	0.991961	0.999461
1024-bit	0.999956	0.998899	0.996627	0.999295



### 4.3 Keyspace analysis and brute force attack

The total number of keys used in the cryptosystem is referred to as the key space. A robust encryption technique should have a large enough key space to overcome brute-force attacks. According to NIST, the minimum key length to resist brute-force attacks is 112 bits; therefore, the key space should be larger than  $2^{112}$  to avoid brute-force attacks [24].

The first phase of the proposed model generates several key lengths depending on the encryption algorithm requirements since it can concatenate more than one key of size  $n$  via the Donald E. Knuths method. For example, at 128, 192, 256, 512, and 1024-bit lengths, the key spaces are  $2^{128}$ ,  $2^{192}$ ,  $2^{256}$ ,  $2^{512}$ , and  $2^{1024}$ , respectively. Table 8 lists the time required to implement the brute-force attack according to each key length. Moreover, the second phase of the proposed model repeats the key generation for three rounds, which increases the degree of complexity. As a final step in the same phase, another key is added of the same length and XOR, which doubles the key space. For example, for a key length of 128 bits, the total key space is  $2^{128} \times 2^{128} \times 2^{128} \times 2^{128} \times 2^{128}$ . In other words, the key space is doubled five times for a key size of  $n$  bits ( $2^{5*n}$ ).

TABLE VIII. BRUTE-FORCE ATTACK TIMES ACCORDING TO KEY LENGTH

Key Length	Years
128-bit	$1.08 * 10^{13}$
192-bit	$1.989 * 10^{38}$
256-bit	$3.67 * 10^{57}$
512-bit	$4.25 * 10^{134}$
1024-bit	$5.698 * 10^{288}$

### 4.4 Analysis of Key Sensitivity

An efficient encryption technique should exhibit high sensitivity to small changes in the secret keys used. This property, known as the avalanche effect, ensures that even minor modifications to the key result in significantly different outputs. Moreover, the key generated from the proposed model is sensitive to the initial parameters, whereas any different values yield various keys. To assess this parameter, various initial parameters were used to generate secret keys of different lengths to yield diverse key values, as shown in Table 9. As shown in the table, the distributions of 0 and 1 in each key are different from the others, which means that any variation in any parameters will yield a key with a different value.

TABLE IX. NUMBERS OF 0 s AND 1 s FOR EACH GENERATED KEY

Key	0's	1's	0's	1's	0's	1's	0's	1's	0's	1's
	128-bit		192-bit		256		512		1024	
1	54	74	87	105	115	141	283	229	516	508
2	66	62	93	99	125	131	255	257	547	477
3	69	59	99	93	128	128	252	260	532	492
4	55	73	100	92	128	128	263	249	528	496

Two keys for each length, key indices 2 and 4, as represented in Table 1, are selected and plotted to show the distribution of values of zeroes and ones. Figures 4 and 5 graphically depict the distribution of key values (0 s and 1 s), which proves the success of the proposed model for random key generation. As shown in the figures, any variations in the values at 0 s and 1 s yield different chart shapes.

128-bit			
No. of 0= 66	No. of 1= 62	No. of 0= 55	No. of 1=73

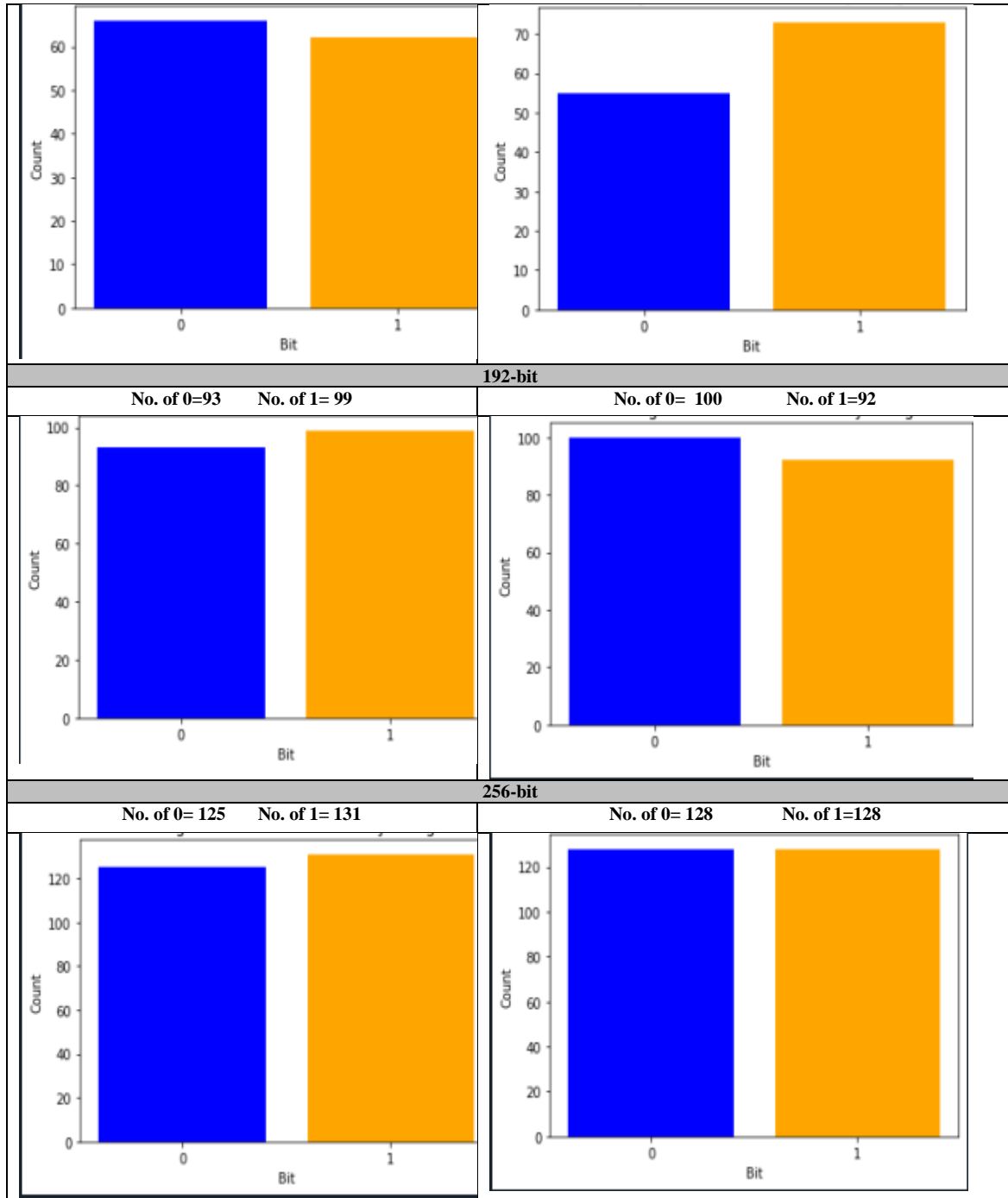


Fig. 4. Various key value distributions for 0 and 1 for 128-bit, 192-bit, and 256-bit keys

512-bit			
No. of 0= 255	No. of 1= 257	No. of 0= 263	No. of 1=249

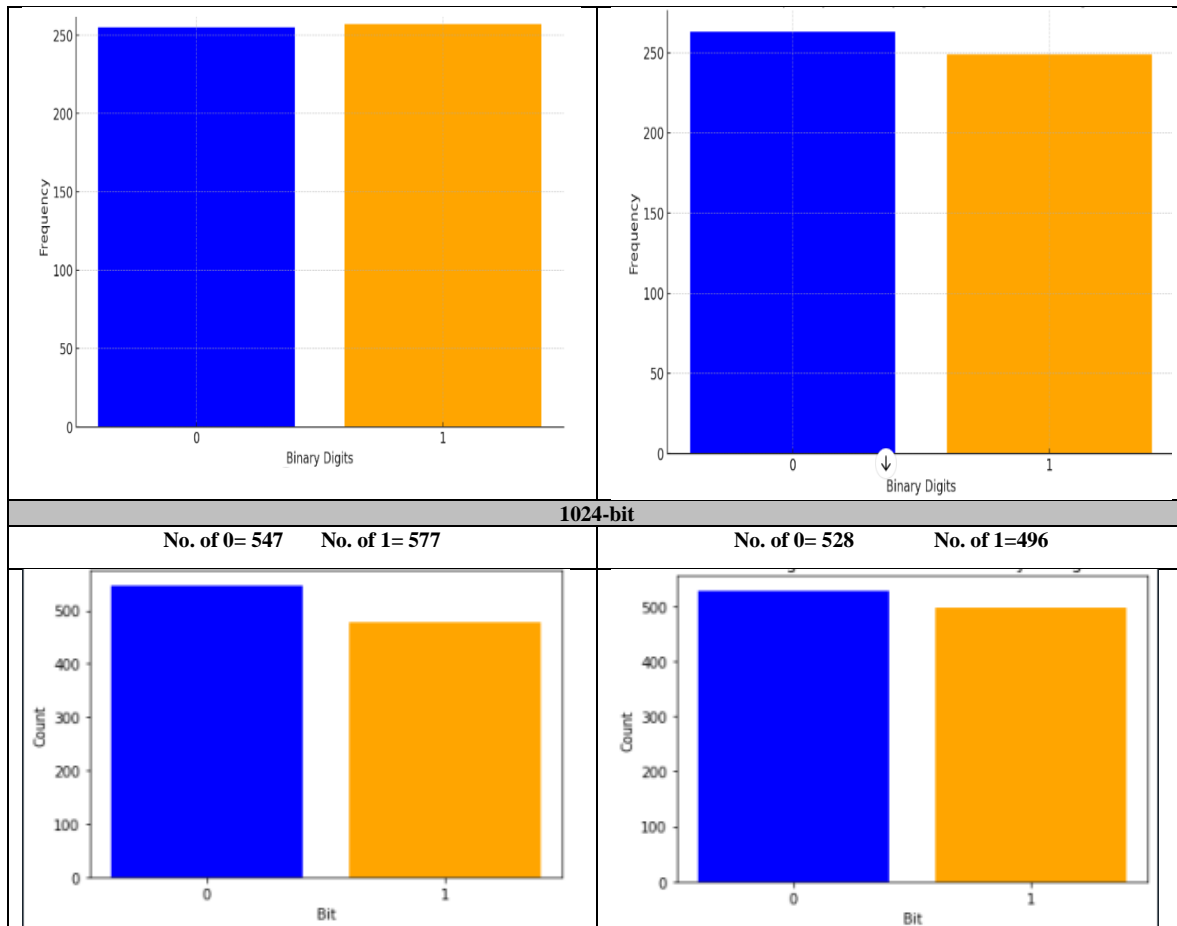


Fig. 5. Key Value Distributions for 0 and 1 for 512-bit and 1024-bit Key Lengths

### 4.5 Resistance Attack Analysis

The assumption is that the cryptanalyst has a complete understanding of the cryptosystem's mechanism, except for the initial seed. The classical four types of attacks used to conclude the source message are listed as follows [25]:

- *Ciphertext-only attack*: the attacker has access only to ciphertexts with no knowledge about either the key or plaintext.
- *Known plaintext* where the attacker is aware of the corresponding plaintext and ciphertext pairs only.
- *Chosen plaintext attack*: The attacker uses the corresponding plaintext-ciphertext pair to analyze the relationship between them, intending to uncover information about the encryption key or algorithm.
- *Chosen Ciphertext attack*: where the attacker has selective access to the decryption system, allowing them to choose specific ciphertexts and obtain the corresponding plaintexts.

The key in the proposed model is sensitive and changes depending on the initial seed variation. This property in the proposed model is resistant to the chosen plaintext attack, which is one of the most common attacks. Moreover, the final operation includes adding another random key to increase the diffusion property to make the cryptosystem more secure and resistant to the aforementioned attacks.

Table 10 presents a comparison with the state-of-the-art methods, which prove the success of the proposed model in enhancing security and improving the key space analysis.

TABLE X. A COMPARISON WITH RELATED WORKS

Ref. No.	Methods	Key Space
[6]	Pseudorandom Sequence Generator (PRSG) based on the concept of modular arithmetic systems with nonintegral numbers	$2^{380}$
[15]	Logistic Map, LFSR, Genetic Algorithm,	$2^{40}$
[17]	Linear Feedback Shift Registers (LFSR) and Linear Congruential Generator (LCG)	$2^{128}$
<b>Propose model</b>	<b>Latin Square Matrix (LSM) and Subtractive Random Number Generator (SRNG)</b>	$2^{640}$

## 5. CONCLUSION

Based on the LSM and SRNG, high-secrecy and random key generation methods were developed. The Donald E. Knuths SRNG algorithm is used to generate variable key sizes as seeds in the first phase. The variable seed value is considered very important since the generated key for every session is not identical or is a one-time pad key. In addition, the variable seed length determines the LSM dimensions and can vary in every session. This property makes it more difficult for the cryptanalyst to guess the length and value of the key. The proposed method randomly selects three rows instead of one and permutes them in depth three times depending on each row's contents. This approach maximizes the randomness and breaks the correlation between bits in each row to increase the diffusion and confusion properties. The final step is to XOR an additional key created randomly to increase complexity and maximize the randomness of the generated key in the proposed model. The NIST, entropy, key analysis, and attack investigation metrics achieved high performance for the results implemented by the proposed model for random key generation.

### Conflicts of interest

The authors declare that they have no conflicts of interest.

### Funding

The author's paper explicitly states that the research project did not receive any funding from institutions or sponsors.

### Acknowledgement

The authors are thankful to the institution for their commitment to supporting scholarly endeavors and creating a conducive research environment.

## References

- [1] Siti Radhiah Binti Megat Azahari1, Sapiee Jamel, Muhammad Faheem Mushtaq, Shamsul Kamal Ahmad Khalid, Kamaruddin Malik Mohamad, Mustafa Mat Deris, “ Tesseract Encryption Algorithm using Latin Square of order 8 (Tea 8) ”, An international journal of advanced computer technology, vol. 9, no. 4, pp. 3617-362, 2020.
- [2] Sura Abed Sarab Hussien, Thair Abed Sarab Hussien, Mustafa Abdulsatar Noori, “ A Proposed Algorithm for Encrypted Data Hiding in Video Stream Based on Frame Random Distribution”. Iraqi Journal of Science, vol. 62, no. 9, pp: 3243-325, 2021,
- [3] Emmanuel Oluwatobi Asani, Godsfavour Biety-Nwanju, Abidemi Emmanuel Adeniyi, Salil Bharany, Ashraf Osman Ibrahim, Anas W. Abulfaraj, Wamda Nagmeldin, “Development of an Image Encryption Algorithm using Latin Square Matrix and Logistics Map”, International Journal of Advanced Computer Science and Applications, vol. 14, no.9, pp. 869-877, 2023, doi: 10.14569/IJACSA.2023.0140991.
- [4] Aber Al-Hyari, Charlie Obimbo, Mua'ad M. Abu-Faraj, Ismail Al-Taharwa, “Generating Powerful Encryption Keys for Image Cryptography With Chaotic Maps by Incorporating Collatz Conjecture”, IEEE Access, vol. 12, pp. 4825-4844, 2024, doi: 10.1109/ACCESS.2024.3349470
- [5] Rusul Mansoor Al-Amri, Dalal N. Hamood, Alaa Kadhim Farhan, “ Theoretical Background of Cryptography ”, Mesopotamian Journal of Cybersecurity. Vol.2023, pp. 7–15, 2023, doi: <https://doi.org/10.58496/MJCS/2023/002>; ISSN: 2958-6542.
- [6] Keshav Sinha, Partha Paul, and Amritanjali, “ An Improved Pseudorandom Sequence Generator and its Application to Image Encryption”, KSII Transactions on Internet and Information Systems. vol. 16, no. 4, pp. 1307-1329, 2022, doi: 10.3837/tiis.2022.04.012.
- [7] D Apdilah, M K Harahap, N Khairina, A M Husein, and M Harahap, “ A Comparison of One Time Pad Random Key Generation using Linear Congruential Generator and Quadratic Congruential Generator”, Journal of Physics, Conference Series, 007, pp. 1-6, 2018, doi: 10.1088/1742-6596/1007/1/012006.
- [8] Ahmed Yousif, Ali H. Kashmar, “ Key Generator to Encryption Images Based on Chaotic Maps”, Iraqi Journal of Science, vol. 60, no. 2, pp. 362-370, 2019, doi: 10.24996/ijs.2019.60.2.16.
- [9] Bashar Adel Esttaifan, “A Modified Vigenère Cipher based on Time and Biometrics features”, Journal of Engineering, vol. 29, no. 6, pp. 128-139, 2023, doi: <https://doi.org/10.31026/j.eng.2023.06.10>.
- [10] Adi A. Maaaita, Hamza A. A. Al Sewadi, “Deterministic Random Number Generator Algorithm for Cryptosystem Keys”. International Journal of Computer and Information Engineering, vol. 9, no. 4, pp. 972-977, 2015.
- [11] Jie Zhou, Nan-Run Zhou, Li-Hua Gong, “ Fast color image encryption scheme based on 3D orthogonal Latin squares and matching matrix”, Optics and Laser Technology, vol., no. 131, pp. 1-14, 2020, doi: [org/10.1016/j.optlastec.2020.106437](https://doi.org/10.1016/j.optlastec.2020.106437).
- [12] Sumit A. Bhagat, Prashant M. Kakade, “ Efficient Route Optimization using Distance Matrices for Generation of Latin Square”, International Journal of Advanced Research in Computer and Communication Engineering, vol. 6, no.1, pp. 380-382, 2017, doi: 10.17148/IJIREICE.2018.6.
- [13] Honglian Shen, Xiuling Shan, Ming Xu, and Zihong Tian, “ A new chaotic image encryption algorithm based on transversals in a Latin square”, Entropy, vol 24, no. 11, pp. 1-19, 2022, doi: 10.3390/e24111574.

- [14] Madhu Sharma, Ranjeet K. Ranjan, Vishal Bharti, “A pseudo-random bit generator based on chaotic maps enhanced with a bit-XOR operation”, *Journal of Information Security and Applications*, vol. 69, 2022, doi: [org/10.1016/j.jisa.2022.103299](https://doi.org/10.1016/j.jisa.2022.103299).
- [15] Fatima Faiz Saleh, Nada Hussein M. Ali, “Generating Streams of Random Key Based on Image Chaos and Genetic Algorithm”, *Iraqi Journal of Science*, vol.63, no. 8, pp.3652-3661, 2022. doi <https://doi.org/10.24996/ij.s.2022.63.8.39%20>.
- [16] Sungju Park, Kyungmin Kim, Keunjin Kim and Choonsung Nam, “Dynamical Pseudo-Random Number Generator Using Reinforcement Learning”, *Applied Science*, vol. 12, no. 7, pp. 1-7, 2022, doi: [org/10.3390/app12073377](https://doi.org/10.3390/app12073377).
- [17] Balsam Abdulkadhim Hameedi, Anwar Abbas Hattab, Muna M. Laftah, “A Pseudo-Random Number Generator Based on New Hybrid LFSR and LCG Algorithm”, *Iraqi Journal of Science*, vol. 63, no. 5, pp. 2230-2242, 2022, doi: <https://doi.org/10.24996/ij.s.2022.63.5.35>.
- [18] Yucheng Chen, Huiqing Huang, Chunming Tang, and Weiming Wei, “A novel adaptive image privacy protection method based on Latin square”, *Nonlinear Dynamics*, Vol. 112, pp:10485–10508, (2024). <https://doi.org/10.1007/s11071-024-09580-1>.
- [19] Guanghui Cao, Yuan Tao, Xiang Liu, and Tianxu Zhang, “Image Encryption Based on a Coined Chaotic System and High-Intensity Encryption Primitives”, *IEEE access*, pp: 92043-92061, 2024. doi:10.1109/ACCESS.2024.3423691
- [20] Marek Sys, Zden Riha, Vashek Matyas, “NIST Statistical Test Suite result interpretation and optimization”, *Conference: SantaCrypt At: Prague 2015*, pp. 14- 17, 2015.
- [21] Mays M. Hoobi, “Multilevel Cryptography Model using RC5, Twofish, and Modified Serpent Algorithms”, *Iraqi Journal of Science*, vol. 65, no. 6, pp. 3434-3450, 2024, doi: <https://doi.org/10.24996/ij.s.2024.65.6.37>.
- [22] Ankita Bisht, Mohit Dua, Shelza Dua, and Priyanka Jaroli, “A Color Image Encryption Technique Based on Bit-Level Permutation and Alternate Logistic Maps”, *J. Intell. Syst.*, vol. 29, no. 1, pp. 1246-1260, 2020, doi: [org/10.1515/jisys-2018-0365](https://doi.org/10.1515/jisys-2018-0365).
- [23] Loay E. George, Enas Kh. Hassan, Sajaa G. Mohammed, Faisal G. Mohammed, “Selective Image Encryption Based on DCT, Hybrid Shift Coding, and Randomly Generated Secret Key”, *Iraqi Journal of Science*, vol. 61, no. 4, pp. 920-935, 2020, doi: [10.24996/ij.s.2020.61.4.2](https://doi.org/10.24996/ij.s.2020.61.4.2).
- [24] Maria Imdad, Sofia Najwa Ramli and Hairulnizam Mahdin, “An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys”, *Symmetry*, vol. 14, no. 3, pp. 1-22, 2022, doi: [org/10.3390/sym14030604](https://doi.org/10.3390/sym14030604).
- [25] Mohit Dua, Aishwarya Wesanekar, Vishwas Gupta, Mayank Bhola, and Shelza Dua, “Differential evolution optimization of intertwining logistic map-DNA based image encryption technique”, *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp.3771–3786, 2020, doi:10.1007/s12652-019-01580-z.
- [26] W. K. Jummar, A. M. Sagheer, and H. M. Saleh, “Authentication System Based on Fingerprint Using a New Technique for ROI selection”, *Babylonian Journal of Artificial Intelligence*, vol. 2024, pp. 102–117, Aug. 2024.
- [27] M. Sheela, R. Suganthi, S. Gopalakrishnan, T. Karthikeyan, K. J. Jyothi, and K. Ramamoorthy, “Secure Routing and Reliable Packets Transmission In MANET Using Fast Recursive Transfer Algorithm”, *BJN*, vol. 2024, pp. 78–87, Jun. 2024.
- [28] S. N. Tambe-Jagtap, “A Survey of Cryptographic Algorithms in Cybersecurity: From Classical Methods to Quantum-Resistant Solutions”, *SHIFRA*, vol. 2023, pp. 43–52, Jun. 2023, doi: [10.70470/SHIFRA/2023/006](https://doi.org/10.70470/SHIFRA/2023/006).