A New Variant Of Hill Cipher Algorithm Using Modified S-Box

Jessie R. Paragas, Ariel M. Sison, Ruji P. Medina

Abstract— Cryptography is used during the exchange of information to guarantee the confidentiality and integrity of data. The hill cipher is a polygraph cipher with a fundamental structure and quick calculations. However, it could be weak with a known-plaintext attack. Another downside is the uncertainty of an invertible key matrix for decryption. In this paper, the process of encryption of plaintext in 128-bit blocks using base 64 conversion, cipher block chaining, and modified substitution box was applied to overcome these weaknesses of the Hill cipher. The proposed method provides improved data security and overcomes the drawbacks of the original hill cipher algorithm. The tests result validates that based on the actual experiment, the security of the ciphertext increases to 64.27% of the avalanche effect score and passes through the randomness test method.

Index Terms— base 64 conversion, cipher block chaining, hill cipher, substitution box

1 Introduction

Cryptography is the science of privacy, data integrity, authentication, and other tasks from unauthorized access [1]. There are two types of encryption in cryptography, either symmetric and asymmetric. The same key (private key) is used in symmetrical encryption for both encryption and decryption, while the sender utilizes a private key other than the recipient's private key in the asymmetric encryption process [2].

Hill suggested a symmetric encryption algorithm in 1929 using invertible matrix. A matrix to transform the plaintext into ciphertext is the basic concept of the algorithm, and the key is the matrix itself [3]. The technique is defined as encryption using the equation $C = KM \bmod R$ where M is a plaintext matrix; C is a matrix of the ciphertext; R is the plaintext value field (in this study, $R = 64$). K is an encryption key, and K must be an invertible matrix. Hill cipher has a high throughput, high velocity, and simple structure, but it succumbs to a known-plaintext attack and the uncertainty of an invertible key matrix for decryption [4].

The Hill cipher algorithm must be improved, especially in the generation of the key matrix, and plaintext encryption process by using a different strategy to enhance ciphertext safety and to decrease chances of a known-plaintext attack.

2 Related Works

Several studies tend to address this problem, such as making the so-called known-plaintext attack harder by raising the numbers of encryption phases to four instead of three but the algorithm's overall output is decreasing [5]. [6] address the search time generating the exact key matrix and reverse key matrix to the key matrix generation by calculating the correct fitness will be more accurate and quicker, but the vulnerability to a known-plaintext attack not being addressed. The classic Playfair algorithm was integrated into the key matrix generation, but restricting the number of characters to 26 encryption and decryption letters reduces security [7].

Many researchers attempted to create the method of Hill cipher and enhance its security. [8] tries to address the decryption issue by setting offset value one where the matrix determinant is zero and offset value -1 if the determinant value is negative; if the key matrix is not invertible. [9] proposed an efficient modification of the Hill cipher using pseudo-random own values and altering the key matrix that resists known plaintext-ciphertext attack by each block dynamically to make the suggested method quicker than any other alteration. The Hill cipher algorithm created a new encryption technique for the process of digital ciphertext values and converting it to ECC points with scalar multiplication, first. This technique improved safety but also reduced computational time because scalar multiplication was used for a long time [10].

- *Jessie R. Paragas is currently pursuing Doctorate degree program in Information Technology in Technological Institute of the Philippines, Quezon City, Philippines. E-mail: alphaphidang@gmail.com*
- *Ariel M. Sison is currently the Dean of School of Computer Studies in Emilio Aguinaldo College, Manila, Philippines. E-mail: ariel.sison@eac.edu.ph*
- *Ruji P. Medina is currently the Dean of Graduate Programs in Technological Institute of the Philippines, Quezon City, Philippines. E-mail: ruji.medina@tip.edu.ph*

3 Proposed Modified Algorithm

The proposed algorithm is a block cipher which consists of three stages such a key matrix generation, plaintext encryption, and decryption. The input is a plaintext of any length. The plaintext is divided into 128-bit block and padding is referred to as the method of adding bits to the final block is also applied. SHA256 password hashing, ciphertext block chaining process [11], substitution box, circular shifting, XOR are utilized to strengthen the ciphertext. The detailed process of each stage is outlined in the next section of the paper.

The program has been written in C++ in this research and subsequently tested through a simulation sequence.

3.1 Key Generation Process

The keys can be generated by random text or data input based on the hashed value of SHA256. The detailed process is as follows:

1. Calculate the SHA256 digest of the key at 32 bytes

output.
2. Divide the digest into four blocks of 8 bytes such as D1, D2, D3, and D4.
3. Compute the result between D1 ⊕ D2 and D3 ⊕ D4.
4. Split the results at 4 bytes and assign it to R1, R2, R3, R4 separately as the rows of Key1.
5. Get the base 64 value of Key1 and assign it to Key2. Key1 and Key2 will be tested if they are invertible before plain text encryption.

Fig. 1 illustrated this method, and Table 1 shows the simulation results for the modified key generation process.

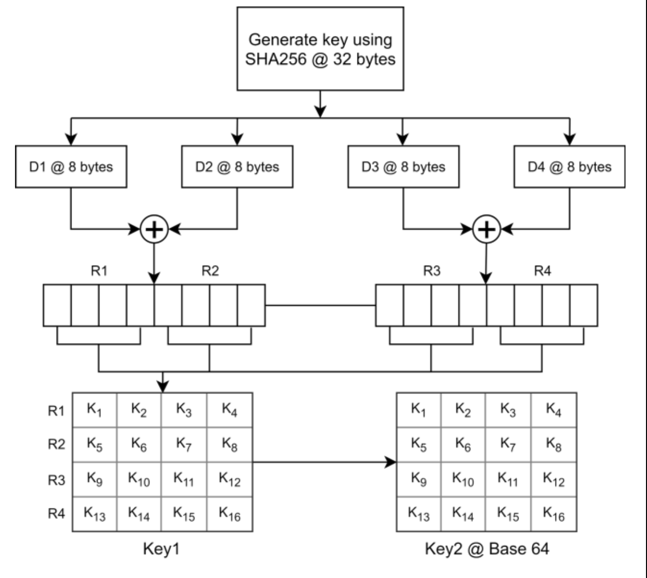


Fig. 1 Modified hill cipher key generation process.

TABLE 1
MODIFIED KEY GENERATION PROCESS

Step	Process	Result
1	Get SHA256 digest of "PASSKEY".	57 10 CF 25 3C 1F 8A 68 3C 38 B2 A5 F6 E6 98 2C 32 4F D9 94 96 35 8D D8 69 A9 39 6C 8B 7E DB 13
2	Divide the digest into four blocks.	D1 = 57 10 CF 25 3C 1F 8A 68; D2 = 3C 38 B2 A5 F6 E6 98 2C; D3 = 32 4F D9 94 96 35 8D D8; D4 = 69 A9 39 6C 8B 7E DB 13
3	Compute the result of XOR and convert to a decimal value.	D1 ⊕ D2: 107 | 40 | 125 | 128 | 202 | 249 | 18 | 68; D3 ⊕ D4: 91 | 230 | 224 | 248 | 29 | 75 | 86 | 203
4	Split the results and assign them to Key1.	R1: 107 40 125 128; R2: 202 249 18 68; R3: 91 230 224 248; R4: 29 75 86 203
5	Get the base 64 value of Key1 and assign it to Key2.	43 40 61 0; 10 57 18 4; 27 38 32 56; 29 11 22 11

3.2 Plaintext Encryption Process

Plaintext encryption requires a 16-byte block of plaintext bits to produce a block of typically same size ciphertext bits. In this process, the total length of the plaintext is divided into a 128-bit block, as shown in Fig. 2. For example, if the given plaintext is "*Cryptography is the ultimate form of non-violent direct action.*", this is divided into 16 characters per block, each of which is 8 bits. The plaintext partition is shown in Table 2. Please note that the remaining characters in Block 4 have a length of 15, which is less than 16. To complete the required 16 characters per block, the system automatically pads one space.

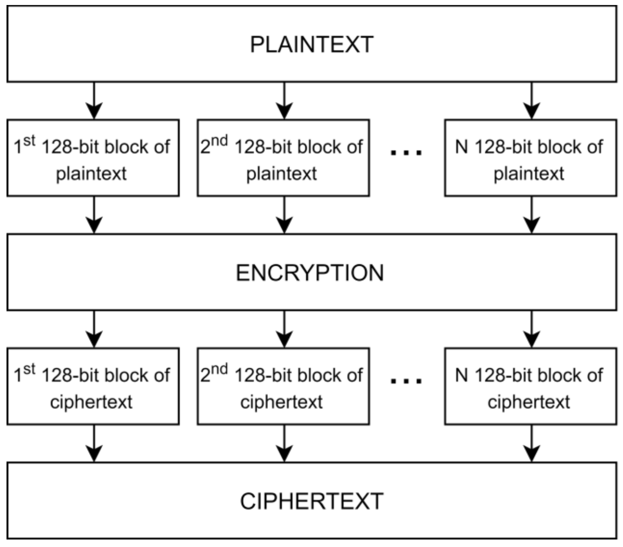
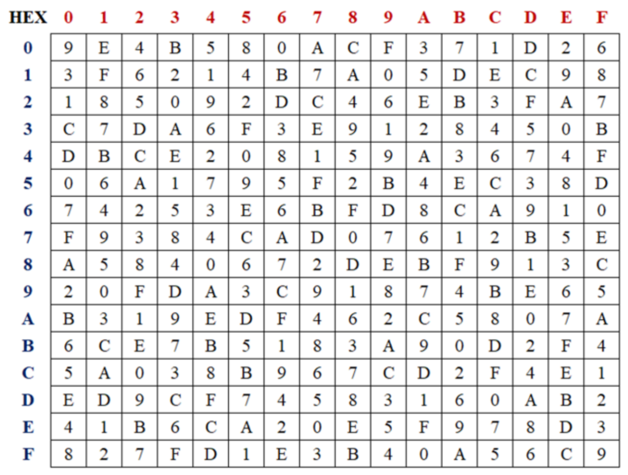

PLAINTEXT
1st 128-bit block of plaintext
2nd 128-bit block of plaintext
...
N 128-bit block of plaintext
ENCRYPTION
1st 128-bit block of ciphertext
2nd 128-bit block of ciphertext
...
N 128-bit block of ciphertext
CIPHERTEXT

Fig. 2 Division of plaintext block at 128-bit.



HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	9	E	4	B	5	8	0	A	C	F	3	7	1	D	2	6
1	3	F	6	2	1	4	B	7	A	0	5	D	E	C	9	8
2	1	8	5	0	9	2	D	C	4	6	E	B	3	F	A	7
3	C	7	D	A	6	F	3	E	9	1	2	8	4	5	0	B
4	D	B	C	E	2	0	8	1	5	9	A	3	6	7	4	F
5	0	6	A	1	7	9	5	F	2	B	4	E	C	3	8	D
6	7	4	2	5	3	E	6	B	F	D	8	C	A	9	1	0
7	F	9	3	8	4	C	A	D	0	7	6	1	2	B	5	E
8	A	5	8	4	0	6	7	2	D	E	B	F	9	1	3	C
9	2	0	F	D	A	3	C	9	1	8	7	4	B	E	6	5
A	B	3	1	9	E	D	F	4	6	2	C	5	8	0	7	A
B	6	C	E	7	B	5	1	8	3	A	9	0	D	2	F	4
C	5	A	0	3	8	B	9	6	7	C	D	2	F	4	E	1
D	E	D	9	C	F	7	4	5	8	3	1	6	0	A	B	2
E	4	1	B	6	C	A	2	0	E	5	F	9	7	8	D	3
F	8	2	7	F	D	1	E	3	B	4	0	A	5	6	C	9

Fig. 3 Modified substitution box.

At this point in the process, each plaintext byte is passed through the substitution box shown in Fig. 3. The substitution box is static to facilitate hardware execution and boost proposed algorithm efficiency. There are 256 bytes in the substitution box. The axis of x (red text) and y (blue text) is the index order. Every byte is represented as a hexadecimal value.

TABLE 2
PLAINTEXT AT 128-BIT PER BLOCK

Block	Plaintext	Length
1	Cryptography is	16 chars
2	the ultimate for	16 chars
3	m of non-violent	16 chars
4	direct action.	15 chars

In Fig. 4 depicts the actual encryption process of a single block of plaintext by getting the first 16 chars converted to binary value at 128-bit of plaintext. The plaintext block will be divided by separating the middle 6-bit part from every byte retaining the enclosing bits to be processed for CBC1. The symmetric key is Key1 while Key2 is being utilized as the encryption key for both pre-processed plaintext and the circularly shifted Key2<1. The result will be XORed before passing to the substitution box. The result from substitution box will be XORed twice with circularly shifted Key2<1 and circularly shifted Key2<1 <<<3 to add more complexity. CBC2 leads to the final ciphertext of the first block as the last phase. The similar encryption process is carried out until the final plaintext block. Table 3 shows the step by step single block plaintext encryption process. Table 4 shows the final ciphertext of each block in hexadecimal value.

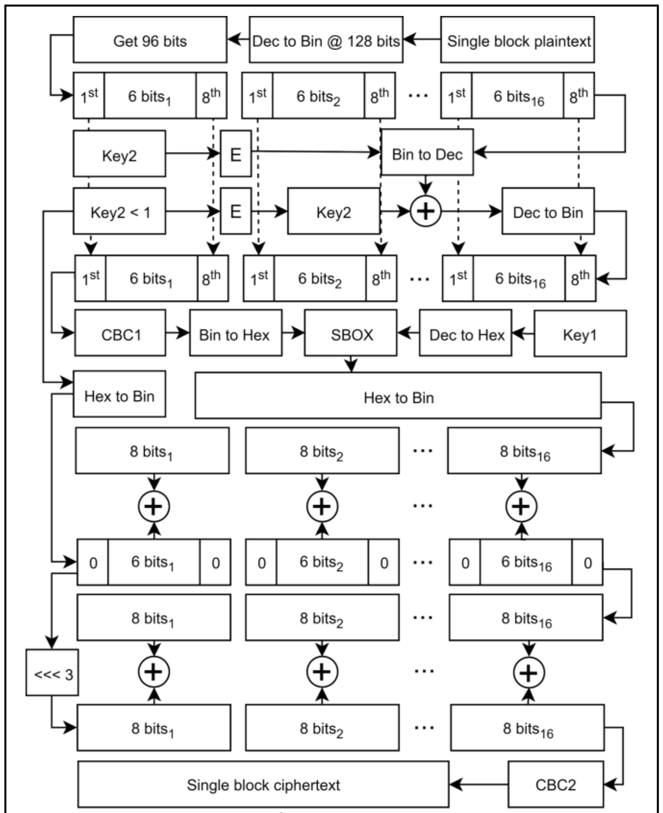


Fig. 4 A single block of the plaintext encryption process.

TABLE 3
PLAINTEXT ENCRYPTION PROCESS

Step	Process	Result
1	Get the ASCII value of Block 1 then convert to a binary value.	01000011 01110010 01111001 01110000 01110100 01101111 01100111 01110010 01100001 01110000 01101000 01111001 00100000 01101001 01110011 00100000
2	Get the middle 6-bit of each value then convert to a decimal value.	NewP = $\begin{array}{\|c\|c\|c\|c\|}\hline 33 & 57 & 60 & 56 \\ \hline 58 & 55 & 51 & 57 \\ \hline 48 & 56 & 52 & 60 \\ \hline 16 & 52 & 57 & 16 \\ \hline\end{array}$
3	Perform C1 = (Key2 * NewP mod 64) ⊕ (Key2 * Key2<1 mod 64).	$\begin{array}{\|c\|c\|c\|c\|}\hline 11 & 3 & 48 & 28 \\ \hline 20 & 57 & 63 & 25 \\ \hline 23 & 13 & 30 & 30 \\ \hline 11 & 30 & 40 & 35 \\ \hline\end{array} \oplus \begin{array}{\|c\|c\|c\|c\|}\hline 27 & 44 & 6 & 45 \\ \hline 14 & 7 & 20 & 18 \\ \hline 21 & 13 & 17 & 32 \\ \hline 25 & 46 & 53 & 0 \\ \hline\end{array}$ C1 = $\begin{array}{\|c\|c\|c\|c\|}\hline 16 & 47 & 54 & 49 \\ \hline 26 & 62 & 43 & 11 \\ \hline 2 & 0 & 15 & 62 \\ \hline 18 & 48 & 29 & 35 \\ \hline\end{array}$
4	Convert the value of C1 to a binary value and perform CBC1 between enclosing bit data from Step 1. Combine C1 and CBC1 then assigned to X.	Enclosing bit data from Step1. **0**100001**1** **0**111001**0** **0**111100**1** **0**111000**0** **0**111010**0** **0**110111**1** **0**110011**1** **0**111001**0** **0**110000**1** **0**111000**0** **0**110100**0** **0**111100**1** **0**010000**0** **0**110100**1** **0**111001**1** **0**010000**0** X = Combine C1 and CBC1 **0**010000**1** **0**101111**1** **0**110110**0** **0**110001**0** **0**011010**0** **0**111110**1** **0**101011**0** **0**001011**0** **0**000010**1** **0**000000**1** **0**001111**1** **0**111110**0** **0**010010**0** **0**110000**1** **0**011101**0** **0**100011**0**
5	Convert X and Key1 to hexadecimal value then split into one-character value. Apply S-box substitution then assign to Z.	Z = Intersection of X and Y axis 2 C 2 C A 0 7 4 3 E 3 E 4 D B 8 0 5 4 4 1 6 3 9 6 F 8 C 1 8 8 1
6	Combine Z by two-character value then convert to a decimal value.	44 44 160 116 62 62 77 184 5 68 22 57 111 140 24 129
7	Convert Z and Key2<1 to a binary value. Enclose Key2<1with 0-bit value. Perform bit XOR between Z and Key2<1 then assign to C2.	Key2<1 = **0**010111**0** **0**010001**0** **0**111011**0** **0**000000**0** **0**010100**0** **0**110011**0** **0**100100**0** **0**001000**0** **0**110110**0** **0**001101**0** **0**000001**0** **0**110001**0** **0**111010**0** **0**010110**0** **0**101100**0** **0**010110**0** C2 = 00000010 00001110 11010110 01110100 00010110 01011000 00000101 10101000 01101001 01011110 00010100 01011011 00011011 10100000 01000000 10101101
8	Apply circular shift <<< 3 to Key2<1. Perform bit XOR between new Key2<1 and C2 then assign to C3.	C3 = 01110011 00011101 01100110 01110101 01010101 01101010 01000101 00101011 00001001 10001110 00000111 01001000 10111010 11000010 10000001 11001100

9	Perform CBC2 then assign to FC then convert to a hexadecimal value as final ciphertext block.	5D E9 BB A6 66 4C 79 CD F1 0B FA 70 D3 7C FE 88
10	Repeat this process until the last block of plaintext. Apply Key2<1, if block index is odd and Key2<<2, if block index is even.	

TABLE 4
RESULT OF PLAINTEXT ENCRYPTION PROCESS

Plaintext	Ciphertext
Cryptography is	5DE9BBA6664C79CDF10BFA70D37CFE88
the ultimate for	DF4D970556423E38E20412AD387F78B9
m of non-violent	E55DE9B99DABA2699DFD784A75B09A5
direct action._	84C6BA0DA035A955512DC47A3B04CC2F

TABLE 5
ENCRYPTION TEST 1 (SAME PLAINTEXT)

Key	Plaintext	No.	Ciphertext
PASSKEY	Crypt**o**graphy is	1	5DE9BBA6664C79CDF10BFA70D37CFE88
	Crypt**p**graphy is	2	A23644599922783C057FFD80DCE10686
	the ulti**m**ate for	1	DF4D970556423E38E2041AD387F78B9
	the ulti**n**ate for	2	274D9705D9BDC1C7B003E15880717B48
	m of non-violent	1	E55DE9B99DABA2699DFD784A75B09A5
	n of non-violent	2	4EA4184C4E5A5B951FF327D30AAF454
	direct actio**n.**_	1	84C6BA0DA035A955512DC47A3B04CC2F
	direct actio**p.**_	2	7B3945F25FCA56AAAED23B85C4F6CE25
Total Average AE %			64.58%

3.3 Decryption Process
The decryption method is performed in the inverse order of the encryption method to recover the plaintext. Based on Fig. 4, Key2 is transformed into the inverse key matrix as Key2^{-1}. The rest of the process could be derived from the encryption method. Moreover, the same substitution box is being used during substitution stage to find the X-axis value.

4 SIMULATION RESULT
The proposed algorithm is tested using the C++ programming language for analyzing the avalanche effect. A small change in the plain text should result in a substantial shift in the ciphertext block [12]. Table 5 presents the encryption test results between two plaintexts that differ by one bit.

Table 6 presents the encryption test results between two plaintexts that differ by one bit and using another key.

TABLE 6
ENCRYPTION TEST 2 (ANOTHER KEY)

Key	Plaintext	Ciphertext
otherkey	Cryptography is	F53AD3F15CC6670ED257D26533C50E11
	the ultimate for	0D6D78844509D8FD613ADC6CA4300F8F
	m of non-violent	599080CE7F6DD78A9684640AA3D502A0
	direct action.	5710633A11C329627026E530F424B357
Total Average AE %		63.95%

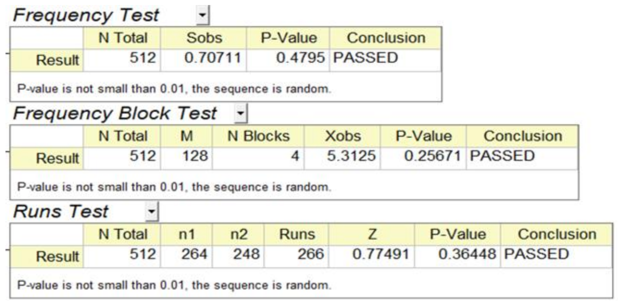
Both tables present a significant test result with an avalanche effect that surpasses the strict avalanche requirement of 50%. Moreover, both encryption tests are performed by changing any letter on any position based on the plus or minus one-bit character difference using the same plaintext.

5 PERFORMANCE AND SECURITY ANALYSIS
5.1 Avalanche Effect
Analyzing the performance of the proposed algorithm based on the avalanche effect, it is compared between the original Hill cipher algorithm and modified existing algorithm. The result is shown in Table 7.

TABLE 7
AVALANCHE EFFECT ANALYSIS (SAME PLAINTEXT)

Modification Position	Avalanche Effect			
	Original Hill Cipher	M. Viswambari & Mani, 2017 [14]	A. Pandey, Pandey, & A. Agarwal, 2018 [10]	Proposed Algorithm
Any position plus/minus one-bit change	8.20%	31.00%	12.61%	64.27%

5.2 Randomness Tests
In cryptography, randomness is essential to make sure that the secret key is random, to protect against attacks, privacy, and anonymity, as well as to ensure unpredictability [13]. Fig. 5 depicts the randomness tests result of the proposed algorithm. The results indicate that the P-value has passed the confidence interval of randomness to ensure the ciphertext is complex.


Frequency Test

	N Total	Sobs	P-Value	Conclusion
Result	512	0.70711	0.4795	PASSED

P-value is not small than 0.01, the sequence is random.

Frequency Block Test

	N Total	M	N Blocks	Xobs	P-Value	Conclusion
Result	512	128	4	5.3125	0.25671	PASSED

P-value is not small than 0.01, the sequence is random.

Runs Test

	N Total	n1	n2	Runs	Z	P-Value	Conclusion
Result	512	264	248	266	0.77491	0.36448	PASSED

P-value is not small than 0.01, the sequence is random.

Fig. 5 Proposed algorithm randomness tests result.

Fig. 6 displays a summary of the outcomes for all random testing conducted. These results show that the proposed algorithm was better than the previous one.

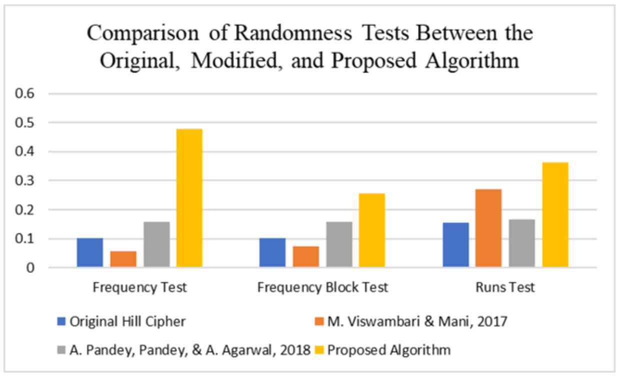


Fig. 6 Randomness tests result between modified existing algorithm.

5.3 Known-Plaintext Attack

For every ciphertext block, this sort of attack requires a thorough analysis. The key is generated using SHA256 password hashing, and its hashed value is divided into four digests and XORed to get the value for Key1. Next step is the conversion of Key1 to its base 64 value for Key2, and this is used to encrypt the plaintext. During encryption, Key1 serves as Y-axis throughout the substitution process; additionally, a ciphertext block chaining method is applied to improve the complexity of the ciphertext. Since the key is generated based on SHA256 digest, it makes the attack more difficult to achieve. Also, the hashed key value serves as the symmetric key to be passed to the receiver along with the ciphertext.

6 Conclusions And Recommendations

A single-bit variation in plain text evaluates the efficiency of the proposed algorithm. The output proves that its security is considerably affected. The tests demonstrate an average 64.27 percent avalanche effect score after a sequence of tests and pass through the randomness test method resulting in improved security compared to the current technique. The key is obtained from the SHA-256 digest and later converted as new key for encryption and decryption process. Also, the entire technique is difficult to crack with a cipher block chaining, as well as substitution box integration. Future work could enhance the security of the cipher by increasing the size of the key, and even the produced key is not invertible, and the encryption and decryption technique is feasible.

Acknowledgment

The authors are grateful for the wisdom, knowledge, and understanding bestowed upon the Almighty God in making this study possible despite every challenge. He deserves all the glory and honor. Moreover, to his family for their continued love, prayer, and moral support in every way.

References

[1] Khalaf, E. T., Mohammed, M. N., & Sulaiman, N. (2016). Iris Template Protection Based on Enhanced Hill Cipher. Proceedings of the 2016 International Conference on Communication and Information Systems - ICCIS '16, 53–57. https://doi.org/10.1145/3023924.3023938

[2] Dawahdeh, Z. E., Yaakob, S. N., & Razif bin Othman, R. (2018). A New Image Encryption Technique Combining Elliptic Curve Cryptosystem with Hill Cipher. Journal of King Saud University - Computer and Information Sciences, 30(3), 349–355. https://doi.org/10.1016/j.jksuci.2017.06.004

[3] Eisenberg, M. (1999). Hill Ciphers and Modular Linear Algebra. Mimeographed Notes, University of Massachusetts, 1–19.

[4] Stallings, W. (2017). Cryptography and Network Security Principles and Practice (7th Editio). Pearson.

[5] Agrawal, K. (2014). Elliptic Curve Cryptography with Hill Cipher Generation for Secure Text Cryptosystem. International Journal of Computer Applications, 106(1), 18–24.

[6] Putera, A., Siahaan, A. P. U. & Rahim, R. (2016). Dynamic Key Matrix of Hill Cipher Using Genetic Algorithm. International Journal of Security and Its Applications, 10(8), 173–180. https://doi.org/10.14257/ijsia.2016.10.8.15

[7] Itagi, A. (2018). Image Encryption Using Orthogonal Hill Cipher. International Journal for Research in Applied Science & Engineering Technology, 6(Ii), 20–27.

[8] Sharma, N. (2014). A Novel Approach to Hill Cipher. International Journal of Computer Applications, 108(11), 975–8887. https://doi.org/10.5120/18958-0285

[9] Mahmoud, A., & Chefranov, A. (2014). Hill Cipher Modification based on Pseudo-Random Eigenvalues. Applied Mathematics & Information Sciences An International Journal, 516(2), 505–516. https://doi.org/http://dx.doi.org/10.12785/amis/080208

[10] A. Pandey, Pandey, S., & A. Agarwal. (2018). Transpoly Hill Cipher - An Improvement Over Traditional Hill Cipher. 9, 276–278. https://doi.org/http://dx.doi org/10.26483/ijarcs.v9i1.520

[11] Kurniawan, D. H., & Munir, R. (2016). Double Chaining Algorithm. 978-1-5090-1636-5/16/$31.00 ©2016 IEEE

[12] Forri, R. (1990). The Strict Avalanche Criterion: Spectral Properties of Boolean Functions. 450–468.

[13] Rukhin, A., Soto, J., & Nechvatal, J. (2010). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications Special Publication800-22R1a. (April).

[14] M. Viswambari, & Mani, K. (2017). Generation of Key Matrix for Hill Cipher using Magic Rectangle. Proceedings - 2nd World Congress on Computing and Communication Technologies, WCCCT 2017, 10(5), 51–54. https://doi.org/10.1109/WCCCT.2016.22