ISSN 2278-3091
Volume 9, No.3, May - June 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse119932020.pdf
https://doi.org/10.30534/ijatcse/2020/119932020

A Modified Polybius Cipher with a New Element-in-Grid Sequencer

Jan Carlo T. Arroyo[1], Allemar Jhone P. Delima[2]
[1]College of Computing Education, University of Mindanao, Davao City, Davao del Sur, Philippines
[2]College of Engineering, Technology and Management, Cebu Technological University-Barili Campus, Cebu, Philippines
jancarlo_arroyo@umindanao.edu.ph[1], allemarjpdjca@yahoo.com[2]

ABSTRACT

This paper modifies the traditional Polybius square with a 5x5 grid through the introduction of a new dynamic substitution-based matrix for ciphertext assignment. The modification is done through the alteration of cell elements, where the ASCII decimal code equivalent of the key is the basis for the element positioning. The proposed enhanced Polybius square called ASCII Code-based Polybius Square Alphabet Sequencer (APSAlpS) is hoped to be used along with other lightweight cryptographic algorithms classified as block ciphers, stream ciphers, and hash functions in solving constrains identified in IoT. The use of the modified Polybius square along with other lightweight ciphers to establish a better IoT security is recommended.

Key words: Cryptography, ciphers, IoT, modified Polybius square

1. INTRODUCTION

The Internet of Things (IoT), as a novel paradigm in the Information Technology arena, has gained a foothold in today's research trend as IoT plays an essential role in human life. The inception of IoT leverages social works as tagging technologies such as radio frequency identification (RFID) and wireless sensor networks (WSN) to wit: wireless fidelity (WiFi), Bluetooth, long-term evolution (LTE), third-generation (3G), global system for mobile communication (GSM), general packet radio service (GPRS), near-field communication (NFC), Zigbee connections and the likes, are now used to connect physical objects almost to everything. With this, a boost in communication mediums and the sharing of information is perceived. In general, IoT interconnects physical objects and living things, bringing both into the sphere of the cyber world. However, as the new technology arises, constraints on the paradigm shift also emerge. Factors such as data security, data integrity, operating environment, communications security, and the likes, hampers the optimal use of IoT. The identified constraints have opened an avenue for a new field, the Lightweight Cryptography. Lightweight ciphers generally classified as block ciphers, hash functions, and stream ciphers are implemented in IoT applications as software implementation cost comes in handy. Further, the flexibility of manufacturing and maintenance using lightweight ciphers is made efficient [1]. Figure 1 shows the application of lightweight cryptography in IoT obtained from [2].

As data protection and the privacy of users is known to be one of the hindrances for wide-scale adoption of IoT [3]–[5], various researchers lobby on the use of cryptography to strengthen security [6], [7]. Cryptography [8], as a known technique used in data communications security for a better IoT implementation, deals with algorithms to ensure data integrity, secured wireless communication systems, and other security services where information transfer takes place between different users [9]–[11]. The bottleneck for optimal use of cryptography relies on the cipher technology used, which varies depending on the problem the researcher wanted to solve [8].

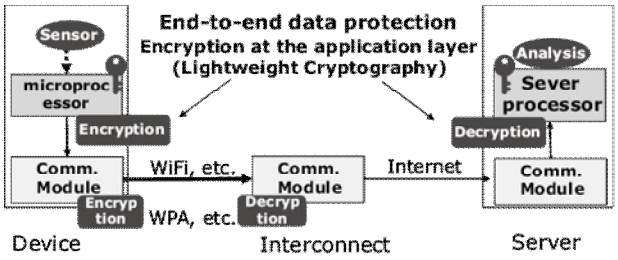

Figure 1: Application of lightweight cryptography in IoT

Algorithms under block ciphers such as AES, Blowfish, 3DES, RC5, and DES [12], along with hash functions and stream ciphers generally called as lightweight ciphers are identified as solutions to IoT constraints, are commonly used to perform the job. However, the use of standalone lightweight ciphers to secure data is not enough. Hybridization of techniques is being used to provide more reliable data and network security protection.

Various classical ciphers such as Railfence cipher [13], [14], Polybius square cipher [15]–[19], Playfair cipher [20]–[22], Permutation cipher [23], Homophonic substitution cipher [24], [25], Hill cipher [26], Grille cipher [27], Four-square cipher [23], Enigma machine cipher [23], Caesar cipher [28]–[30], Base64 cipher [31]–[33], Affine cipher [23], [34]–[36], and ADFGVX cipher [23], [37]–[39] to name some, are being incorporated with block ciphers, among others, to solve the constraints in IoT. Among the classical ciphers, the Polybius Cipher, known as the Polybius square, is identified as one of the commonly used methods for such purpose [40]–[42]. However, the Polybius square has tradeoffs and is easy to crack with frequency analysis due to the simplicity of element distribution scheme in the grid [43]. The root cause of the problem is in the structure of how digraphs are produced as ciphertext. Therefore, there is a

need to introduce a new digraph identification scheme; thus, this study. The modified Polybius square is hoped to add strength to the security, when combined with lightweight ciphers, as a medium to establish more secure IoT systems.

2. RELATED LITERATURE

The domain of IoT spans from healthcare, smart cities, connected cars, wearables, and smart homes, among others [1]. Cryptography [10] plays a vital role in the implementation of IoT. This obscuring technique is known to protect data from various industries [44]–[47]. Communications security is ensured through the use of two basic ciphers, the substitution [43] and transposition [35], whose bottleneck for an optimal implementation relies on the cipher algorithm used for encipherment and decipherment process.

Polybius Cipher, being one of the commonly used cipher techniques in the literature, is continuously being improved for better security performance. The following subsections are the modifications made on the Polybius square.

2.1 Embedding Data Crypted with Extended Shifting Polybius Square Supporting Turkish Character Set

An extended Polybius Square presented in [15] works by integrating Turkish characters in a 10x7 grid. The extended version also introduced a dynamic matrix wherein elements of a matrix shift values depending on a numeric value. The study proposes unique substitution values every time the matrix components shift. The extended Polybius matrix with Turkish characters is presented in Table 1.

Table 1: Polybius square with Turkish characters

	01	02	03	04	05	06	07
01	A	B	C	Ç	D	E	F
02	G	Ğ	H	I	İ	J	K
03	L	M	N	O	Ö	P	R
04	S	Ş	T	U	Ü	V	Y
05	Z	Q	X	W	1	2	3
06	4	5	6	7	8	9	0
07	.	,	:	;	+	-	*
08	/		!	“	#	$	%
09	&	=	<	>	?	@	[
10	]	\	__	(	)	{	}

First, a shifting value is set to generate a new matrix, which is used for encryption. For instance, the shift value is ‘0052’; thus, a new grid is created, as shown in Table 2.

Table 2: Shifted Polybius square

	01	02	03	04	05	06	07
01	“	#	$	%	&	=	<
02	>	?	@	[	]	\	_
03	(	)	{	}	A	B	C
04	Ç	D	E	F	G	Ğ	H
05	I	İ	J	K	L	M	N
06	O	Ö	P	R	S	Ş	T
07	U	Ü	V	Y	Z	Q	X
08	W	1	2	3	4	5	6
09	7	8	9	0	.	,	:
10	;	+	-	*	/		!

To encrypt a value, each plaintext character is converted to its corresponding ciphertext in a 4-digit format using the matrix. For instance, ‘AZ GİT’ is converted into ‘030507051006040505020607’ as presented in Table 3. The decryption process is done by dividing the ciphertext into groups of 4-digit numbers and then matching each group to the matrix.

Table 3: Encryption using shifting Polybius square

Plaintext	A	Z	(space)	G	İ	T
Ciphertext	0305	0705	1006	0405	0502	0607

The proposed method was successfully implemented in a steganographic algorithm. Since ciphertext values generated by the scheme can range from 0101 to 1007, these can be applied to modify the least-significant bit (LSB) of an image file to embed encrypted messages. Findings from the Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measurement (SSIM) tests show that there are no obvious alterations and distortions in the file.

2.2 Implementation of Nihilist Cipher Algorithm in Securing Text Data with Md5 Verification

The use of a 5x5 Polybius square in encrypting messages presented in [16] differs in the arrangement of the characters from the traditional scheme. The study also proposed the use of a key for increased security. Based on the example given in Table 4, letters are arranged column-wise, from top to bottom, then left to right.

Table 4: Polybius square using Nihilist cipher

	1	2	3	4	5
1	A	F	L	Q	V
2	B	G	M	R	W
3	C	H	N	S	X
4	D	I	O	T	Y
5	E	K	P	U	Z

To perform encryption, the plaintext and the key are converted as ciphertext using the given Polybius matrix. Each resulting digraph from the plaintext and key is summed together to generate the final encrypted message. As an example, the plaintext ‘THINGS’ is encoded using the secret key ‘DEVICE’. The first character ‘T’ from the plaintext is translated as 44, whilst the first character ‘D’ from the secret key is translated as 41. Adding both digraphs, the ciphertext for ‘T’ and ‘D’ becomes 85 (44+41=85). After converting each of the characters, the final ciphertext value for the whole plaintext is ‘858357755385,’ as shown in Table 5.

Table 5: Encryption using Nihilist cipher

Plaintext	T	H	I	N	G	S
Ciphertext	44	32	42	33	22	34
Key	D	E	V	I	C	E
Ciphertext	41	51	15	42	31	51
Final Ciphertext	85	83	57	75	53	85

For decryption, the secret key must be provided. First, the key is translated into ciphertext using the Polybius Square. Each translated digraph is paired with the corresponding digraphs from the ciphertext. To retrieve the plaintext, the digraphs from the key are deducted from the digraphs of the ciphertext. The process is repeated until it reaches the end of the plaintext.

A test was performed to ensure data integrity using the MD5 hash function. Results show that despite encryption and decryption processes, the intended message remained intact and genuine.

3. METHODOLOGY

3.1 Traditional Polybius Square

The Polybius Cipher uses a square grid composed of 5 rows and 5 columns. In this matrix, the letters of the English alphabet are placed in alphabetical order from left to right and top to bottom. Each cell in the matrix is identified according to its relative index in the grid represented by the combination of the row and column number, as shown in Table 6.

Table 6: Traditional Polybius square matrix

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Since no key is needed for this scheme, encryption and decryption using this technique are made easy. To encrypt a message, each character from the string is located in the matrix to retrieve its respective coordinate based on the value of the intersection of row and columns. The accumulated coordinate values will represent the ciphertext. For example, encrypting the word 'INTERNET' results to '2433441542331544' where the character 'I' is 24, 'N' is 33, and so forth. The encryption result using the traditional Polybius square is presented in Table 7.

Table 7: Encryption using traditional Polybius square

Plain Text	I	N	T	E	R	N	E	T
Position	1	2	3	4	5	6	7	8
Ciphertext	24	33	44	15	42	33	15	44

The decryption process is done in a reverse manner. Each digraph is compared to the grid to retrieve its corresponding plaintext value. For example, the encoded message '2433441542331544' is translated as 'INTERNET,' as shown in Table 8.

Table 8: Decryption using traditional Polybius square

Ciphertext	24	33	44	15	42	33	15	44
Position	1	2	3	4	5	6	7	8
Plain Text	I	N	T	E	R	N	E	T

3.2 Enhanced Polybius Square

The enhanced Polybius square called ASCII Code-based Polybius Square Alphabet Sequencer (APSAlpS) uses the traditional Polybius Square matrix composed of letters 'a' to 'z.' However, in this method, the use of a secret key is required. With the key, the elements inside the matrix are reorganized by shifting cells based on its ASCII decimal values. A cell shifting is done for every plaintext character encoded; thus, a new matrix is generated for each iteration. Similar plaintext letters may not have the same ciphertext value, thus ensuring that the encoded message generated by the modified technique is always dynamic and, therefore, more complicated to break using cryptanalysis.

For the APSAlpS to work, it is required to identify a plaintext and a secret key. Every character in the plaintext is matched with a character from the key based on its relative position. The process is repeated until it reaches the last character of the plaintext. Next, the ASCII decimal code value of each character of the key is retrieved. The ASCII decimal code value of the character serves as the basis to perform the number of shifts required to reorder the elements in the matrix. After every shift, the new matrix is used to retrieve the equivalent ciphertext value. The message 'LIGHTWEIGHT' with the key 'IOT' and its relative ASCII decimal codes is shown in Table 9.

Table 9: Plaintext and key with ASCII equivalent

Plaintext	L	I	G	H	T	W	E	I	G	H	T
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84

With the given values, the original matrix performs 76 shifts to the right, as shown in Table 10 below. The new matrix is used to retrieve the encoding value of the first character of the plaintext wherein the character 'C' is encrypted as the value 24, as shown in Table 11.

Table 10: New matrix after 1st cell shift

	1	2	3	4	5
1	I/J	K	L	M	N
2	O	P	Q	R	S
3	T	U	V	W	X
4	Y	Z	A	B	C
5	D	E	F	G	H

Table 11: 1st character encryption using APSAlpS

Plaintext	L	I	G	H	T	W	E	I	G	H	T
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84
Cipher Text	24										

To encrypt the next character, a new matrix is generated again by shifting the elements to the right 73 times. As a result, the character 'I' is encoded as 31, as represented in Tables 12 and 13.

Table 12: New matrix after 2nd cell shift

	1	2	3	4	5
1	T	U	V	W	X
2	Y	Z	A	B	C
3	D	E	F	G	H
4	I/J	K	L	M	N
5	O	P	Q	R	S

Table 13: 2nd character encryption using APSAlpS

Plaintext	L	I	G	H	T	W	E	I	G	H	T
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84
Cipher Text	24	31									

The procedure is reiterated up to the length of the plaintext. The final encrypted value at the end of the process is now regarded as '24 31 43 42 22 44 55 23 35 34 14', as shown in Table 14.

The variation in the encrypted values between the use of traditional Polybius Square and the modified Polybius Square using similar plaintexts is presented in Table 15. Based on the results, it is apparent that the ciphertext generated by both methods are completely different from one another. The table also depicts that the common ciphertext codes 22, 23, 24, 31, and 44 of both traditional and modified methods have different values in the latter. This denotes that even if the

encryption values share the same code, they may not have equivalent plaintext value, thus making the cipher difficult to break using frequency analysis.

Table 14: Encrypted values using APSAlpS

Plaintext	L	I	G	H	T	W	E	I	G	H	T
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84
Cipher Text	24	31	43	42	22	44	55	23	35	34	14

Table 15: Comparison between methods

Traditional Polybius Square											
Plaintext	L	I	G	H	T	W	E	I	G	H	T
Cipher Text	31	24	22	23	44	52	15	24	22	23	44
APSAlpS											
Plaintext	L	I	G	H	T	W	E	I	G	H	T
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84
Cipher Text	24	31	43	42	22	44	55	23	35	34	14

The secret key is required for decrypting an encoded message. Each digraph from ciphertext must be matched in with a character from the key. The process is repeated until the last digraph of the ciphertext is reached. Next, the ASCII decimal code value of each character of the key is retrieved. The retrieved ASCII decimal code value serves as the basis to perform the number of shifts required to reorder the elements in the matrix. After every shift, the new matrix is used to retrieve the code equivalent for every digraph. The process is repeated until all digraphs are converted to their respective plaintext values. The encrypted message '24 31 43 42 22 44 55 23 35 34 14' with the secret key 'IOT' and its corresponding ASCII decimal codes is shown in Table 16.

Table 16: Decrypted values using APSAlpS

Ciphertext	24	31	43	42	22	44	55	23	35	34	14
Key	I	O	T	I	O	T	I	O	T	I	O
ASCII Value	76	73	71	72	84	87	69	73	71	72	84
Plaintext	L	I	G	H	T	W	E	I	G	H	T

To check how both traditional and modified methods work for strings with repetitive letters, the phrase 'TALLAHASSEE' is encrypted. The simulation results are shown in Table 17.

Table 17: Comparison using repeated letters

Traditional Polybius Square											
Plaintext	T	A	L	L	A	H	A	S	S	E	E
Cipher Text	44	11	31	31	11	23	11	43	43	15	15
APSAlpS											
Plaintext	T	A	L	L	A	H	A	S	S	E	E
Key	F	L	F	L	F	L	F	L	F	L	F
ASCII Value	85	83	85	83	85	83	85	83	85	83	85
Cipher Text	34	52	12	13	33	51	24	12	52	25	15

Obvious patterns are evident for the encrypted values produced by using the traditional method wherein the repeating values 11 for 'A,' 31 for 'L,' '43' for S, and 15 for 'E' are observed several times. However, looking closely at the ciphertext produced by the APSAlpS, it is apparent that no two same plaintext values are encrypted identically to the traditional method. Also, taking out similar ciphertext codes from the modified technique does not necessarily equate to having the same plaintext values such that 12 could mean 'L' or 'S,' and 52 could mean 'A' or 'S.' The results of the modified method also show that similar plaintext values may have variating codes such that 'S' can be represented as either 12 or 52, as against in the traditional method wherein the same characters share the same code.

3.3 Evaluation Methods

One way of predicting the value of the ciphertext is done using frequency analysis [48]. To test the output of the proposed method for frequency analysis, a given plaintext is encrypted and then submitted as input to an online digraphs-digits-only frequency analysis tool [49]. The following text was used for testing:

"IhavemyselffullconfidencethatifalldotheirdutyifnothingisneglectedandifthebestarrangementsaremadeastheyarebeingmadeweshallproveourselvesonceagainabletodefendourIslandhometorideoutthestormofwarandtooutlivethemenaceoftyrannyifnecessaryforyearsifnecessaryaloneAtanyratethatiswhatweareg oingtotrytodoThatistheresolveofHisMajestysGovernmenteverymanofthemThatisthewillofParliamentandthenationTheBritishEmpireandtheFrenchRepubliclinkedtogetherintheircauseandintheirneedwilldefendtothedeaththeirnativesoilaidingeach otherlikegoodcomradestotheutmostoftheirstrengthEventhoughlargetractsofEuropeandmanyoldandfamousStateshavefallenormayfallintothegripoftheGestapoandalltheodiousapparatusofNaziruleweshallnotflagorfailWeshallgoontotheendweshallfightinFranceweshallfightontheseasandoceansweshallfightwithgrowingconfidenceandgrowingstrengthintheairweshalldefendourIslandwhatevertheCostmaybeweshallfightonthebeaches weshallfightonthelandinggroundsweshallfightinthefieldsandinthestreetsweshallfightinthehillsweshallneversurrenderandevenifwhichIdonotforamomentbelievethisIslandoralargepartofitweresubjugatedandstarvingthenourEmpirebeyondtheseasarmedandguardedbytheBritishFleetwouldcarryonthestruggleuntilinGodsgoodtimetheNewWorldwithallitspowerandmightstepsforthtotherescueandtheliberationoftheold"

The modified method was also assessed by recording its execution time. A simple program was developed in an i7-7700HQ 2.80GHz 16GB RAM 4GB VRAM laptop computer and Python 3.

4. RESULTS AND DISCUSSION

Upon performing the evaluation methods, the simulation results showed that the APSAlpS offers more layers of security with the use of a secret key and dynamically generated matrices.

The frequency analysis result of the encoded text using the traditional Polybius Square is presented in Table 18. Based on the findings, the top 10 digraphs 15, 44, 11, 33, 23, 24, 34, 42, 31and 43 have the most frequency in usage. If these values are converted using the traditional method, the decrypted values would be E, T, A, N, H, I, O, R, L, S, respectively. The results follow the study of [50], which presented the most frequent letters used in the Latin alphabet, wherein the top 10 are the same as the findings. Simulation results prove that a monoalphabetic cipher such as the Polybius square is certainly susceptible to frequency analysis and, therefore, simple to break [8], [10], [14], [51].

Table 18: Frequency analysis results using the traditional Polybius square

Digraphs	Frequency	%
15	169×	13.28%
44	122×	9.58%
11	107×	8.41%
33	93×	7.31%
23	87×	6.83%
24	85×	6.68%
34	85×	6.68%
42	75×	5.89%
31	73×	5.73%
43	72×	5.66%
14	58×	4.56%
21	43×	3.38%
22	41×	3.22%
52	28×	2.2%
45	26×	2.04%
32	25×	1.96%
13	22×	1.73%
54	18×	1.41%
51	15×	1.18%
12	13×	1.02%
35	13×	1.02%
25	2×	0.16%
55	1×	0.08%

The frequency analysis result of the ciphertext produced by the modified Polybius Square is presented in Table 19. Findings show that there is a minimal disparity in the frequency of each digraph. This means that the use of the digraphs is almost equally distributed among themselves. Consequently, decrypting the encoded message through frequency analysis can be challenging and may take time as every digraph or code may represent a number of plaintext values. For example, the most frequent digraph 55 could mean any of the characters depending on the cycle of transformations the matrix had undergone. This, therefore, makes the modified Polybius Square not susceptible to frequency analysis.

Table 19: APSAlpS frequency analysis result

Digraphs	Frequency	%
55	68×	5.34%
53	66×	5.18%
12	61×	4.79%
45	61×	4.79%
32	56×	4.4%
23	54×	4.24%
42	53×	4.16%
25	53×	4.16%
35	53×	4.16%
31	53×	4.16%
44	51×	4.01%
51	51×	4.01%
54	50×	3.93%
41	50×	3.93%
21	49×	3.85%
43	49×	3.85%
11	49×	3.85%
52	48×	3.77%
13	47×	3.69%
14	45×	3.53%
15	45×	3.53%
22	44×	3.46%
24	40×	3.14%
34	39×	3.06%
33	38×	2.99%

In terms of the execution time of the modified method as compared to the traditional Polybius Square, results reveal that the former has higher execution time with 0.0031s as compared to the latter with 0.0005s. This finding can be attributed to the fact that the modified scheme generates a new matrix for every character to be encrypted. In contrast, the traditional technique only uses a static grid, hence requiring fewer processes done, and in turn, a lower execution time. The indexed simulation results for the execution time are shown in Table 20.

Table 20: Execution time results

String:	MISSISSIPPI
Length:	11 characters
APSAlpS Key:	US
Traditional P.S. Execution Time	0.003192900000000165 s
APSAlpS Execution Time	0.0005627999999999744 s

5. CONCLUSION AND RECOMMENDATION

This study modifies the 5x5 Polybius square by dynamically shifting the elements in the grid determined through the keys' ASCII code. Simulation results revealed that the proposed method is more secure against the unmodified Polybius cipher. Since the modification strengthens the cipher capability of the Polybius square, it is suggested that the modified cipher be used along with other lightweight cryptographic algorithms in increasing the security in IoT domains.

REFERENCES

[1] D. Sehrawat and N. S. Gill, "Lightweight Block Ciphers for IoT based applications: A Review," *Int. J. Appl. Eng. Res.*, vol. 13, no. 5, pp. 2258–2270, 2018.

[2] O. Toshihiko, "Lightweight cryptography applicable to various IoT devices," *NEC Tech. J.*, vol. 12, no. 1, pp. 67–71, 2017.

[3] Q. Xu, P. Ren, H. Song, and Q. Du, "Security enhancement for IoT communications exposed to eavesdroppers with uncertain locations," *IEEE Access*, vol. 4, pp. 2840–2853, 2016. https://doi.org/10.1109/ACCESS.2016.2575863

[4] A. Kaur, "Internet of Things (IoT): Security and Privacy Concerns," *Int. J. Eng. Sci. Res. Technol.*, vol. 5, no. 5, pp. 161–165, 2016.

[5] Kan-Siew-Leong, P. L. R. Chze, A. K. Wee, E. Sim, and K. E. May, "A multi-factors security key generation mechanism for IoT," in *International Conference on Ubiquitous and Future Networks, ICUFN*, 2017, pp. 1019–1021. https://doi.org/10.1109/ICUFN.2017.7993953

[6] H. Tao, M. Z. A. Bhuiyan, A. N. Abdalla, M. M. Hassan, J. M. Zain, and T. Hayajneh, "Secured Data Collection with Hardware-Based Ciphers for IoT-Based Healthcare," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 410–420, 2019. https://doi.org/10.1109/JIOT.2018.2854714

[7] H. Shin, H. K. Lee, H. Y. Cha, S. W. Heo, and H. Kim, "IoT Security Issues and Light Weight Block Cipher," in *1st International Conference on Artificial Intelligence in Information and Communication*, 2019, pp. 381–384.

[8] W. Stallings, *Cryptography and Network Security Principles and Practices*. Prentice Hall, 2015.

[9] B. N. Rao, D. Tejaswi, K. A. Varshini, K. P. Shankar,

and B. Prasanth, "Design of modified AES algorithm for data security," *Int. J. Technol. Res. Eng.*, vol. 4, no. 8, pp. 1289–1292, 2017.

[10] O. Reyad, "Cryptography and Data Security: An Introduction," 2018.

[11] S. N. Kumar, "Review on Network Security and Cryptography," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 8, no. 6, p. 21, 2018.

[12] A. Abd and S. Al-Janabi, "Classification and Identification of Classical Cipher Type Using Artificial Neural Networks," *J. Eng. Appl. Sci.*, vol. 14, no. 11, pp. 3549–3556, 2019.

[13] A. Banerjee, M. Hasan, and H. Kafle, "Secure Cryptosystem Using Randomized Rail Fence Cipher for Mobile Devices," in *Intelligent Computing - Proceedings of the Computing Conference*, 2019, pp. 737–750.

[14] A. P. U. Siahaan, "Rail Fence Cryptography in Securing Information," *Int. J. Sci. Eng. Res.*, vol. 7, no. 7, pp. 535–538, 2016.

[15] H. B. Macit, A. Koyun, and M. E. Yüksel, "Embedding Data Crypted With Extended Shifting Polybius Square Supporting Turkish Character Set," *BEU J. Sci.*, vol. 8, no. 1, pp. 234–242, 2019. https://doi.org/10.17798/bitlisfen.455126

[16] E. V. Haryannto, M. Zulfadly, Daifiria, M. B. Akbar, and I. Lazuly, "Implementation of Nihilist Cipher Algorithm in Securing Text Data with Md5 Verification," *J. Phys. Conf. Ser.*, vol. 1361, no. 012020, 2019.

[17] G. Manikandan, P. Rajendiran, R. Balakrishnan, and S. Thangaselvan, "A Modified Polybius Square Based Approach for Enhancing Data Security," *Int. J. Pure Appl. Math.*, vol. 119, no. 12, pp. 13317–13324, 2018.

[18] M. Maity, "A Modified Version of Polybius Cipher Using Magic Square and Western Music Notes," *Int. J. Technol. Res. Eng.*, vol. 1, no. 10, pp. 1117–1119, 2014.

[19] C. Kumar, S. Dutta, and S. Chakraborty, "A Hybrid Polybius-Playfair Music Cipher A Hybrid Polybius-Playfair Music Cipher," *Int. J. Multimed. Ubiquitous Eng.*, vol. 10, no. 8, pp. 187–198, 2015.

[20] R. Deepthi, "A Survey Paper on Playfair Cipher and its Variants," *Int. Res. J. Eng. Technol.*, vol. 4, no. 4, pp. 2607–2610, 2017. https://doi.org/10.14257/ijmue.2015.10.8.19

[21] M. Syahrizal, M. Murdani, S. D. Nasution, M. Mesran, R. Rahim, and A. P. U. Siahaan, "Modified Playfair Cipher Using Random Key Linear Congruent Method," in *International Seminar: Research, Technology and Culture*, 2017.

[22] R. Rahim and A. Ikhwan, "Cryptography Technique with Modular Multiplication Block Cipher and Playfair Cipher," *Int. J. Sci. Res. Sci. Technol.*, vol. 2, no. 6, pp. 71–78, 2016.

[23] M. S. Hossain Biswas *et al.*, "A systematic study on classical cryptographic cypher in order to design a smallest cipher," *Int. J. Sci. Res. Publ.*, vol. 9, no. 12, pp. 507–11, 2019.

[24] M. Shumay and G. Srivastava, "PixSel: Images as book cipher keys an efficient implementation using partial homophonic substitution ciphers," *Int. J. Electron. Telecommun.*, vol. 64, no. 2, pp. 151–158, 2018.

[25] G. Zhong, "Cryptanalysis of Homophonic Substitution Cipher Using Hidden Markov Models," 2016.

[26] P. E. Coggins and T. Glatzer, "An Algorithm for a Matrix-Based Enigma Encoder from a Variation of the Hill Cipher as an Application of 2 × 2 Matrices," *Primus*, vol. 30, no. 1, 2020.

[27] J. Liu *et al.*, "The Reincarnation of Grille Cipher: A Generative Approach," *Cryptogr. Secur.*, pp. 1–27, 2018.

[28] A. Singh and S. Sharma, "Enhancing Data Security in Cloud Using Split Algorithm, Caesar Cipher, and Vigenere Cipher, Homomorphism Encryption Scheme," in *Emerging Trends in Expert Applications and Security*, 2019, vol. 841, pp. 157–166. https://doi.org/10.1007/978-981-13-2285-3_20

[29] I. Gunawan, Sumarno, H. S. Tambunan, E. Irawan, H. Qurniawan, and D. Hartama, "Combination of Caesar Cipher Algorithm and Rivest Shamir Adleman Algorithm for Securing Document Files and Text Messages," *J. Phys. Conf. Ser.*, vol. 1255, 2019.

[30] D. Gautam, C. Agrawal, P. Sharma, M. Mehta, and P. Saini, "An Enhanced Cipher Technique Using Vigenere and Modified Caesar Cipher," in *2nd International Conference on Trends in Electronics and Informatics, ICOEI 2018*, 2018. https://doi.org/10.1109/ICOEI.2018.8553910

[31] F. Anwar, E. H. Rachmawanto, C. A. Sari, and de Rosal Ignatius Moses Setiadi, "StegoCrypt Scheme using LSB-AES Base64," in *International Conference on Information and Communications Technology, ICOIACT 2019*, 2019, pp. 85–90.

[32] A. R. Pathak, S. Deshpande, and M. Panchal, "A Secure Framework for File Encryption Using Base64 Encoding," in *Computing and Network Sustainability*, vol. 75, Springer Singapore, 2019, pp. 359–366.

[33] R. Rahim, S. Sumarno, M. T. Multazam, S. Thamrin, and S. H. Sumantri, "Combination Base64 and GOST algorithm for security process," *J. Phys. Conf. Ser.*, vol. 1402, 2019. https://doi.org/10.1088/1742-6596/1402/6/066054

[34] M. Maxrizal and B. D. Aniska Prayanti, "Application of Rectangular Matrices: Affine Cipher Using Asymmetric Keys," *CAUCHY –Jurnal Mat. Murni dan Apl.*, vol. 5, no. 4, pp. 181–185, 2019.

[35] T. M. Aung and N. N. Hla, "A Complex Polyalphabetic Cipher Technique Myanmar Polyalphabetic Cipher," in *2019 International Conference on Computer Communication and Informatics, ICCCI 2019*, 2019, pp. 1–9.

[36] O. Laia, E. M. Zamzami, Sutarman, F. G. N. Larosa, and A. Gea, "Application of Linear Congruent Generator in Affine Cipher Algorithm to Produce Dynamic Encryption," *J. Phys. Conf. Ser.*, vol. 1361, no. 1, pp. 1–6, 2019.

[37] I. B. Venkateswarlu and J. Kakarla, "Password security by encryption using an extended ADFGVX cipher," *Int. J. Inf. Comput. Secur.*, vol. 11, no. 4–5, pp. 510–523, 2019. https://doi.org/10.1504/IJICS.2019.101938

[38] R. Mahendran and K. Mani, "Generation of Key

Matrix for Hill Cipher Encryption Using Classical Cipher," *2nd World Congr. Comput. Commun. Technol. WCCCT 2017*, pp. 51–54, 2017.

[39] G. Lasry, I. Niebel, N. Kopal, and A. Wacker, "Deciphering ADFGVX messages from the Eastern Front of World War I," *Cryptologia*, vol. 41, no. 2, pp. 101–136, 2017. https://doi.org/10.1080/01611194.2016.1169461

[40] P. Kumar and S. B. Rana, "Development of modified AES algorithm for data security," *Optik (Stuttg).*, vol. 127, no. 4, pp. 2341–2345, 2016.

[41] J. S. Prasath, U. Ramachandraiah, and G. Muthukumaran, "Modified Hardware Security Algorithms for Process Industries Using Internet of Things," *J. Appl. Secur. Res.*, pp. 1–14, 2020.

[42] L. R and K. M, "Enhancing the security of AES through small scale confusion operations for data communication," *Microprocess. Microsyst.*, vol. 75, p. 103041, 2020.

[43] F. Patel and M. Farik, "A New Substitution Cipher - Random-X," *Int. J. Sci. Technol. Res.*, vol. 5, no. 11, pp. 125–128, 2015.

[44] K. Al Harthy, F. Al Shuhaimi, and K. K. J. Al Ismaily, "The upcoming Blockchain adoption in Higher-education: Requirements and process," in *4th MEC International Conference on Big Data and Smart City, ICBDSC 2019*, 2019, pp. 1–5. https://doi.org/10.1109/ICBDSC.2019.8645599

[45] P. Kuppuswamy, R. Banu, and N. Rekha, "Preventing and securing data from cyber crime using new authentication method based on block cipher scheme," in *2nd International Conference on Anti-Cyber Crimes, ICACC 2017*, 2017, pp. 113–117.

[46] R. Beck, M. Avital, M. Rossi, and J. B. Thatcher, "Blockchain Technology in Business and Information Systems Research," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 381–384, 2017. https://doi.org/10.1007/s12599-017-0505-1

[47] S. Cho, Y. Jeong, and C. Oh, "An efficient cryptography for healthcare data in the cloud environment," *J. Converg. Inf. Technol.*, vol. 8, no. 3, pp. 63–69, 2018.

[48] J. F. Dooley, *History of Cryptography and Cryptanalysis*. 2018.

[49] "Frequency Analysis Tool," Retrieved from *https://www.dcode.fr/frequency-analysis.* .

[50] G. Grigas and A. Juškevičienė, "Letter Frequency Analysis of Languages Using Latin Alphabet," *Int. Linguist. Res.*, vol. 1, no. 1, pp. 18–31, 2018. https://doi.org/10.30560/ilr.v1n1p18

[51] D. Kahn, *Codebreakers*. Macmillan and Sons, 1967.