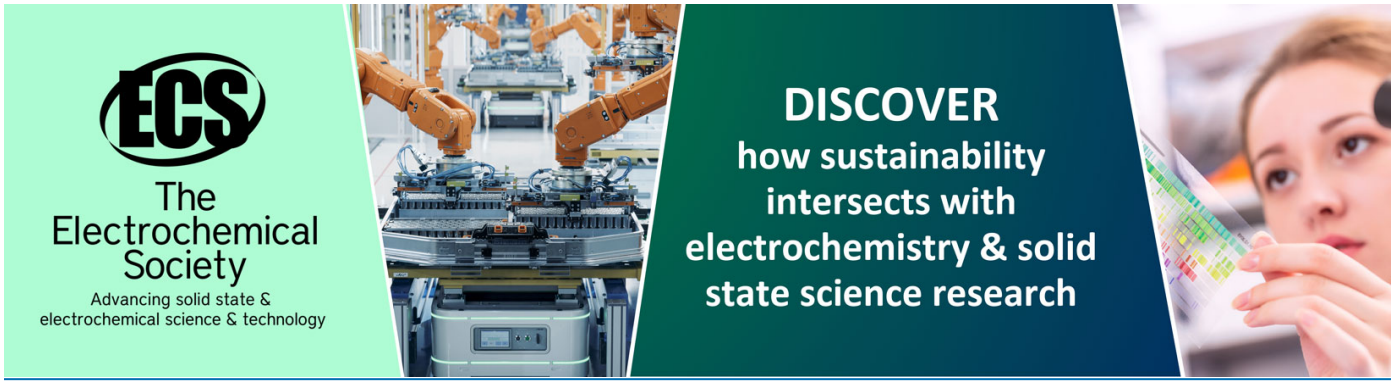
Secure a Transaction Activity with Base64 Algorithm and Word Auto Key Encryption Algorithm



IOP Conf. Series: Journal of Physics: Conf. Series **1028** (2018) 012053 doi:10.1088/1742-6596/1028/1/012053

Secure a Transaction Activity with Base64 Algorithm and Word Auto Key Encryption Algorithm

Heri Nurdiyanto[1], Robbi Rahim[2*], Ansari Saleh Ahmar[3], Muhammad Syahril[4], Muhammad Dahria[5] and Herlina Ahmad[6]

[1]Department of Informatics Engineering, STMIK Dharma Wacana, Metro Lampung, Indonesia
[2]School of Computer and Communication Engineering, Universiti Malaysia Perlis, Kubang Gajah, Malaysia
[3]Department of Statistics, Universitas Negeri Makassar, Makassar, Indonesia,
[4]Department of Information System, STMIK Triguna Dharma, Medan, Indonesia
[5]Department of Information System, STMIK Triguna Dharma, Medan, Indonesia
[6]Department of Mathematic Education, Universitas Al Asyariah Mandar, Polewali Mandar, Indonesia

*usurobbi85@zoho.com

Abstract. Security in a transaction activity is crucial, in digital communication that irresponsible parties can tap all objects sent in the form of bits, the use of cryptographic algorithms such as Base64 and Word Auto Key Encryption can be a solution that produces a secure ciphertext, base64 algorithm which can encode all objects such as text, image, and audio files into text form and the result is re-secured with Word Auto Key Encryption algorithm, the results of this research get better security by combining the two algorithms.

1. Introduction

Data is crucial for maintaining the confidentiality of information, especially containing information that is vital and can only be known by certain parties only [1]–[5], especially if the delivery is arranged through the public network, and if the data is not secured first, it will be very easily tapped by others and easily known contents by parties who do not have the authority [6]–[9]. One way that is used to secure data security is to use a cryptographic system that is by providing the contents of the information (plaintext) into content that is not recognized through the process of encryption (encipher) and to recover the original information are using decryption process (decipher) [3]. Encryption is offered at the time of delivery by converting the original data into secret data, while decryption is performed on acceptance by converting the confidential data into the original data[1], [4], [7], so the original data cannot be known by unauthorized parties [2], [10].

Securing the data in this research using base64 algorithm and word auto key encryption. Base64 is a generic term for a number of similar coding schemes that encode binary data and translate into a base64 representation[11]. The base64 term comes from certain MIME encoding content, and the Base64 algorithm uses one of the modern block encryption algorithms in the form of bit operation and also Base64 algorithm is easier to implement than other algorithms[11], [12]. Word Auto Key Encryption Algorithm is an algorithm that uses a 128-bit key, and a 256 x 32-bit table, word auto key encryption uses XOR, AND, OR and Shift Right operations[6]. Cryptography is the study of how to

secure information, and this security is implementing by encrypting and decrypting the information with a special key[4]. Information that has not experienced the encryption process called plaintext, while the information that has undergone the encryption process is called ciphertext[1], [4]. Cryptographers have created various cryptographic algorithms, but attempts to solve are not the least of which bring success.

This research tries to combine base64 and word auto key encryption algorithm to secure transaction activity process in various fields that require data transmission in the process like network, SMS, email and so on.

2. Methodology

A. Base64 Algorithm

The Base64 algorithm is one of the algorithms for Encoding and Decoding an object into ASCII format, which is meant for the base number 64 or one of the methods used to encode the binary data[11], [12]. Base64 Commonly used in various applications such as e-mail via MME, XML data, or for URL encoding purposes.

The encoding principle is to select a collection of 64 printable characters, so data can be stored and transferred across media designed to handle text data, another use of Base64 encoding is to obfuscate or randomize data[13]. Base64 encryption schemes are usually also used when a password is needed against binary data designed to handle text-shaped data, which is intended to preserve data during transmission to a server. The characters generated by this Base64 transformation consist of A.Z, a..z and 0..9, and attached to last two characters symbolized + and/and one character equal to (=) used for adjustment and fitting Binary data or the term is applied to as filler fitting[11]–[13]. The character of the symbol to be generated will depend on the running algorithm process. Base64 cryptography is widely used in the internet world as a medium data format to send data, this is because the result of Base64 form are plaintext, then this information will be much easier to send, compared to the format of information in the form of binary[11], [14], for the index value of the base64 algorithm can be seen in the table 1 below:

Table 1. Base64 Index Value

Index	Value	Index	Value	Index	value
0	A	28	c	56	4
1	B	29	d	57	5
2	C	30	e	58	6
3	D	31	f	59	7
4	E	31	g	60	8
5	F	33	h	61	9
6	G	34	i	62	+
7	H	35	j	63	-
8	I	36	k		
9	J	37	l		
10	K	38	m		
11	L	39	n		
12	M	40	o		
13	N	41	p		
14	O	42	q		
15	P	43	r		
16	Q	44	s		
17	R	45	t		
18	S	46	u		
19	T	47	v		
20	U	48	w		

21	V	49	x
22	W	50	y
23	X	51	z
24	Y	52	0
25	Z	53	1
26	a	54	2
27	b	55	3

The Base64 Encoding[11] technique is simple, and then the process are like below:

1. Split the bytes string into by-3 bytes.
2. Combine 3 bytes into 24 bits. With a note of 1 bytes = 8 bits, so 3x8 = 24 bits.
3. Then 24 bits are stored in a buffer (put together) broken into 6 bits, it will produce four fractions.
4. Each fraction is changed into a decimal value, where the maximum value of 6 bits is 63.
5. Finally, make the decimal values become indexes to select the character of the preparation of Base64, and the maximum is 63 or index to 64, and so on until the end of the string bytes, we want to convert. If it turns out in the encoding process there is a residual divider, then add as the remaining sequester character =. So sometimes on Base64 will appear one or two characters (=).

B. Word Auto Key Encryption Algorithm

Word Auto Key Encryption algorithm is one of the commercially used stream cipher algorithms, and the algorithm was invented by David Wheeler in 1993[6]. Word Auto Key Encryption algorithm uses a 128 bit key with a 256 x 32-bit table, in the process, this algorithm uses XOR, AND, OR and Shift Right operations[6]. There are several main processes of word auto key encryption algorithm, such as:

1. The method of forming table S-Box (Substitution Box).
2. Key forming process.
3. Encryption and decryption process.

The essence of the Auto Key Encryption Word algorithm lies in the process of forming S-Box tables and forming keys. The S-Box table used is flexible and varies for each round performed[6].

3. Result and Discussion

One of the activities that can be secured is the delivery of short messages sent via SMS service, SMS services contained in the phone/smartphone do not have encryption and decryption process, although at the time of sending SMS on GSM network have been secured, but in principle security that used also be public to facilitate performed data communication, but the message only changed into PDU form and this can be encoded to get back the original message.

Figure 1. SMS transmission with no encryption process

One the solution that can be done is before the SMS sent first must be encoded by using base64 algorithm and then re-encrypted with Word Auto Key Encryption algorithm, assume a message "RobbiRahim" with length ten character, illustration as in Table 2 below:

Table 2. Base64 Example Process

Index	1	2	3	4	5	6	7	8	9	10

Char	R	o	b	b	i	R	a	h	i	m
Dec	81	111	98	98	105	81	97	104	105	109

The calculations from a message can be seen in table 3.

Table 3. ASCII and Binary Message

Index 1	R	Index 6	R
ASCII	81	ASCII	81
Binary	01010010	Binary	01010010
Index 2	o	Index 7	a
ASCII	111	ASCII	97
Binary	01101111	Binary	01100001
Index 3	b	Index 8	h
ASCII	98	ASCII	104
Binary	01100010	Binary	01101000
Index 4	b	Index 9	i
ASCII	98	ASCII	105
Binary	01100010	Binary	01101001
Index 5	i	Index 10	m
ASCII	105	ASCII	109
Binary	01101001	Binary	01101101

Binaries acquired after being combined:
010100100110111101100010011000100110100101010010011000010110100001101001011011 01

Table 4. 6 Bit Binary

Index	Binary 6 Bit	Decimal
1	010100	20
2	100110	38
3	111101	61
4	100010	34
5	011000	24
6	100110	38
7	100101	37
8	010010	18
9	011000	24
10	010110	22
11	100001	33
12	101001	41
13	011011	27
14	000001	1

The ciphertext of the “RobbiRahim” message based on table 1 base64 above is **Um9iYmlSYWhpbB**; this ciphertext will then be re-encrypted using Word Auto Key Encryption algorithm, for the first step is to form S-Box table from Key= ‘Base64 and WAKE ’, the process as:

a. First step, initialitation TT[0] ... TT[7]

TT[0] = 726A8F3B
TT[1] = E69A3B5C
TT[2] = D3C71FE5
TT[3] = AB3C73D2
TT[4] = 4D3A8EB3
TT[5] = 0396D6E8
TT[6] = 3D4C2F7A
TT[7] = 9EE27CF3

b. *Split key into 4 sub keys hexadecimal:*
T[0] = K[0] = 42617365
T[1] = K[1] = 36342061
T[2] = K[2] = 6E642057
T[3] = K[3] = 414B4520

c. *Third step do shift right and XOR process, this process perform from n = 4 until n = 255:*
n = 4
-> X = T[0] + T[3] = 42617365 + 414B4520 = 83ACB885
-> X >> 3 (Shift Right 3 bit) = 83ACB885 >> 3 = 10759710
X AND 7 = 83ACB885 AND 7(10) = 5
T[4] = X >> 3 XOR TT[X AND 7] = 10759710 XOR TT[5] = 13E341F8

n = 5
-> X = T[1] + T[4] = 36342061 + 13E341F8 = 4A176259
-> X >> 3 (Shift Right 3 bit) = 4A176259 >> 3 = 0942EC4B
X AND 7 = 4A176259 AND 7(10) = 1
T[5] = X >> 3 XOR TT[X AND 7] = 0942EC4B XOR TT[1] = EFD8D717

Loop the process until n = 255; and the result as below:
n = 255
-> X = T[251] + T[254] = 3148D98F + 7082B66D = A1CB8FFC
-> X >> 3 (Shift Right 3 bit) = A1CB8FFC >> 3 = 143971FF
X AND 7 = A1CB8FFC AND 7(10) = 4
T[255] = X >> 3 XOR TT[X AND 7] = 143971FF XOR TT[4] = 5903FF4C

d. *Fourth step, loop process n=0 until n = 22*
n = 0
T[0] = T[0] + T[89] = 42617365 + 136780E9 = 55C8F44E

n = 1
T[1] = T[1] + T[90] = 36342061 + 3D571FAE = 738B400F

n = 2
T[2] = T[2] + T[91] = 6E642057 + A8088957 = 166CA9AE

n = 3
T[3] = T[3] + T[92] = 414B4520 + F55A1F06 = 36A56426

n = 4
T[4] = T[4] + T[93] = 13E341F8 + 9FFA4F0E = B3DD9106

n = 5
T[5] = T[5] + T[94] = EFD8D717 + 5690A364 = 46697A7B

Do this fourth step until n = 22, and the result as below:

n = 22
T[22] = T[22] + T[111] = E10EDE0D + 35D6AA81 = 16E5888E

e. *Fifth step, set variable value as:*
X = 65F71165
Z = T[59] OR 01000001 = B32F75D8 OR 01000001 = B32F75D9
Z = Z AND FF7FFFFF = B32F75D9 AND FF7FFFFF = B32F75D9
X = X AND FF7FFFFF = 65F71165 AND FF7FFFFF = 18A6873E

f. *Six step, for n = 0 and n = 255 do this process*
n = 0
X = (18A6873E AND FF7FFFFF) + B32F75D9 = CB55FD17
T[0] = 55C8F44E] AND 00FFFFFF XOR CB55FD17 = CB9D0959

n = 1
X = (CB55FD17 AND FF7FFFFF) + B32F75D9 = 7E8572F0
T[1] = 738B400F] AND 00FFFFFF XOR 7E8572F0 = 7E0E32FF

n = 2
X = (7E8572F0 AND FF7FFFFF) + B32F75D9 = 3134E8C9
T[2] = 166CA9AE] AND 00FFFFFF XOR 3134E8C9 = 31584167

Do this six step process until all value n = 255, last value from function will get:

n = 255
X = (656CEA65 AND FF7FFFFF) + B32F75D9 = 189C603E
T[255] = 5903FF4C] AND 00FFFFFF XOR 189C603E = 189F9F72

g. *Seven step, set variable value*
T[256] = T[0] = CB9D0959
X = X AND 255(10) = 189C603E AND 255(10) = 0000003E

h. *Eight step, this the final process for S-Box tables for n = 0 and n = 255, the process as below:*
n = 0
Temp = T[62] XOR X AND 255 = 2599B191 XOR 0000003E AND 255 = 000000AF
T[0] = T[175] = 28C25F77
T[62] = T[1] = 7E0E32FF

n = 1
Temp = T[63] XOR X AND 255 = D8A6E64C XOR 0000003E AND 255 = 00000072
T[1] = T[114] = 81F15257
T[62] = T[2] = 31584167

n = 2
Temp = T[60] XOR X AND 255 = BF3CECC0 XOR 0000003E AND 255 = 000000FE
T[2] = T[254] = 65EE5C08

T[62] = T[3] = E4C13A84

Do this process until n = 255, and the result as:
n = 255
Temp = T[193] XOR X AND 255 = 365CC015 XOR 0000003E AND 255 = 0000002B
T[255] = T[43] = F1B13AFD
T[62] = T[256] = CB9D0959

After the S-Box table process is finished, next process is to determine the key and change into four sub keys, the process as below:

Key= 'Base64 and WAKE ' keys are change in hex = 426173653634206 16E642057414B4520 split the key into four groups and input into A (0), B (0), C (0) and D (0).
A(0) = 42617365
B(0) = 36342061
C(0) = 6E642057
D(0) = 414B4520
The key for encryption and decryption process is obtained by three rounds rotation, for each round could be done up to n rotation, here is the process of rotation

ROUND KEY 1

FunctionM (A[0],D[0]) = FunctionM(42617365,414B4520) = (42617365 + 414B4520)>>8 XOR T[(42617365 + 414B4520) AND 255(10)] = 83ACB885>>8 XOR T[133] = 0083ACB8 XOR BEBF1225 = BE3CBE9D
A[1] = BE3CBE9D

FunctionM (B[0],A[1]) = FunctionM(36342061,BE3CBE9D) = (36342061 + BE3CBE9D)>>8 XOR T[(36342061 + BE3CBE9D) AND 255(10)] = F470DEFE>>8 XOR T[254] = 00F470DE XOR B395F1C2 = B361811C
B[1] = B361811C

FunctionM (C[0],B[1]) = FunctionM(6E642057,B361811C) = (6E642057 + B361811C)>>8 XOR T[(6E642057 + B361811C) AND 255(10)] = 21C5A173>>8 XOR T[115] = 0021C5A1 XOR EE40E81D = EE612DBC
C[1] = EE612DBC

FunctionM (D[0],C[1]) = FunctionM(414B4520,EE612DBC) = (414B4520 + EE612DBC)>>8 XOR T[(414B4520 + EE612DBC) AND 255(10)] = 2FAC72DC>>8 XOR T[220] = 002FAC72 XOR 3CA2DDFE = 3C8D718C
D[1] = 3C8D718C

ROUND KEY 2

FunctionM(A[1],D[1]) = FunctionM(BE3CBE9D,3C8D718C) = (BE3CBE9D + 3C8D718C)>>8 XOR T[(BE3CBE9D + 3C8D718C) AND 255(10)] = FACA3029>>8 XOR T[41] = 00FACA30 XOR FF2C2946 = FFD6E376
A[2] = FFD6E376

FunctionM(B[1],A[2]) = FunctionM(B361811C,FFD6E376) = (B361811C + FFD6E376)>>8 XOR T[(B361811C + FFD6E376) AND 255(10)] = B3386492>>8 XOR T[146] = 00B33864 XOR 82F77395 = 82444BF1
B[2] = 82444BF1

FunctionM(C[1],B[2]) = FunctionM(EE612DBC,82444BF1) = (EE612DBC + 82444BF1)>>8 XOR T[(EE612DBC + 82444BF1) AND 255(10)] = 70A579AD>>8 XOR T[173] = 0070A579 XOR 45C37F08 = 45B3DA71
C[2] = 45B3DA71

FunctionM(D[1],C[2]) = FunctionM(3C8D718C,45B3DA71) = (3C8D718C + 45B3DA71)>>8 XOR T[(3C8D718C + 45B3DA71) AND 255(10)] = 82414BFD>>8 XOR T[253] = 0082414B XOR C0BB1404 = C039554F
D[2] = C039554F

ROUND KEY 3

FunctionM(A[2],D[2]) = FunctionM(FFD6E376,C039554F) = (FFD6E376 + C039554F)>>8 XOR T[(FFD6E376 + C039554F) AND 255(10)] = C01038C5>>8 XOR T[197] = 00C01038 XOR FF2C2946 = FFEC397E
A[3] = FFEC397E

FunctionM(B[2],A[3]) = FunctionM(82444BF1,FFEC397E) = (82444BF1 + FFEC397E)>>8 XOR T[(82444BF1 + FFEC397E) AND 255(10)] = 8230856F>>8 XOR T[111] = 00823085 XOR C0BB1404 = C0392481
B[3] = C0392481

FunctionM(C[2],B[3]) = FunctionM(45B3DA71,C0392481) = (45B3DA71 + C0392481)>>8 XOR T[(45B3DA71 + C0392481) AND 255(10)] = 05ECFEF2>>8 XOR T[242] = 0005ECFE XOR 2A60E894 = 2A65046A
C[3] = 2A65046A

FunctionM(D[2],C[3]) = FunctionM(C039554F,2A65046A) = (C039554F + 2A65046A)>>8 XOR T[(C039554F + 2A65046A) AND 255(10)] = EA9E59B9>>8 XOR T[185] = 00EA9E59 XOR 99252C5A = 99CFB203
D[3] = 99CFB203

Key = D[3] = 99CFB203

For encryption process with Word Auto Key Encryption algorithm as function below.
PlainText: 'Um9iYmlSYWhpbB'
ASCII Code of 'U' = 55
ASCII Code of 'm' = 6D
ASCII Code of '9' = 39
ASCII Code of 'i' = 69
ASCII Code of 'Y' = 59
ASCII Code of 'm' = 6D
ASCII Code of 'l' = 6C
ASCII Code of 'S' = 53
ASCII Code of 'Y' = 59
ASCII Code of 'W' = 57

ASCII Code of 'h' = 68
ASCII Code of 'p' = 70
ASCII Code of 'b' = 62
ASCII Code of 'B' = 42

Plain Text (hexadecimal) = 556D3969596D6C53595768706242
Key = FF9FD2EF

Cipher Text = Plain Text XOR Key
55 XOR 99 = CC = 'Ì'
6D XOR CF = A2 = '¢'
39 XOR B2 = 8B = '‹'
69 XOR 03 = 6A = 'j'
59 XOR 99 = C0 = 'À'
6D XOR CF = A2 = '¢'
6C XOR B2 = DE = 'Þ'
53 XOR 03 = 50 = 'P'
59 XOR 99 = C0 = 'À'
57 XOR CF = 98 = '˜'
68 XOR B2 = DA = 'Ú'
70 XOR 03 = 73 = 's'
62 XOR 99 = FB = 'û'
42 XOR CF = 8D = '•'

Ciphertext = Ì¢‹jÀ¢ÞPÀ˜Úsû

The result of the ciphertext dispatched to the recipient of the message, the next process is to decrypt the ciphertext by using word auto key encryption algorithm.

Ciphertext = Ì¢‹jÀ¢ÞPÀ˜Úsû

Ascii Code 'Ì' = CC
Ascii Code '¢' = A2
Ascii Code '‹' = 8B
Ascii Code 'j' = 6A
Ascii Code 'À' = C0
Ascii Code '¢' = A2
Ascii Code 'Þ' = DE
Ascii Code 'P' = 50
Ascii Code 'À' = C0
Ascii Code '˜' = 98
Ascii Code 'Ú' = DA
Ascii Code 's' = 73
Ascii Code 'û' = FB
Ascii Code '•' = 8D

Ciphertext (Hexadecimal) = CCA28B6AC0A2DE50C098DA73FB8D

Key = FF9FD2EF

Plain Text = Cipher Text XOR Key

CC XOR 99 = 55 = 'U'
A2 XOR CF = 6D = 'm'
8B XOR B2 = 39 = '9'
6A XOR 03 = 69 = 'i'
C0 XOR 99 = 59 = 'Y'
A2 XOR CF = 6D = 'm'
DE XOR B2 = 6C = 'l'
50 XOR 03 = 53 = 'S'
C0 XOR 99 = 59 = 'Y'
98 XOR CF = 57 = 'W'
DA XOR B2 = 68 = 'h'
73 XOR 03 = 70 = 'p'
FB XOR 99 = 62 = 'b'
8D XOR CF = 42 = 'B'

Plaintext= Um9iYmlSYWhpbB

After getting the plaintext of Word Auto Key Encryption algorithm decryption process next is to do decoding with a Base64 algorithm to get original message, so the security process that occurs in the process of sending SMS messages can see in the following figure.

Figure 2. SMS transmission with encryption Base64 & WAKE

4. Conclusion

Base64 algorithm is an algorithm that can be used for all objects such as text, images and other objects, message security with the base64 algorithm and Word Auto Key Encryption algorithms produce better ciphertext, one of the advantages of using WAKE algorithm is that the key can be *n* value, the higher *n* then the resulting ciphertext is also better.

References

[1] A. Putera, U. Siahaan, and R. Rahim, "Dynamic Key Matrix of Hill Cipher Using Genetic Algorithm," *Int. J. Secur. Its Appl.*, vol. 10, no. 8, pp. 173–180, Aug. 2016.
[2] R. Rahim and A. Ikhwan, "Study of Three Pass Protocol on Data Security," *Int. J. Sci. Res.*, vol. 5, no. 11, pp. 102–104, Nov. 2016.
[3] 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.
[4] R. Rahim, "128 Bit Hash of Variable Length in Short Message Service Security," *Int. J. Secur. Its Appl.*, vol. 11, no. 1, pp. 45–58, Jan. 2017.
[5] H. Nurdiyanto and R. Rahim, "Enhanced pixel value differencing steganography with government standard algorithm," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, 2017, pp. 366–371.
[6] L. Legito and R. Rahim, "SMS Encryption Using Word Auto Key Encryption," *nternational J. Recent Trends Eng. Res.*, vol. 3, no. 1, pp. 251–256, 2017.
[7] D. Nofriansyah and R. Rahim, "COMBINATION OF PIXEL VALUE DIFFERENCING

ALGORITHM WITH CAESAR ALGORITHM FOR STEGANOGRAPHY," *Int. J. Res. Sci. Eng.*, vol. 2, no. 6, pp. 153–159, 2016.

[8] E. Hariyanto and R. Rahim, "Arnold's Cat Map Algorithm in Digital Image Encryption," *Int. J. Sci. Res.*, vol. 5, no. 10, pp. 1363–1365, Oct. 2016.

[9] R. Rahim, "Man-in-the-middle-attack prevention using interlock protocol method," *ARPN J. Eng. Appl. Sci.*, vol. 12, no. 22, pp. 6483–6487, 2017.

[10] R. Rahim, M. Dahria, M. Syahril, and B. Anwar, "Combination of the Blowfish and Lempel-Ziv-Welch algorithms for text compression," *World Trans. Eng. Technol. Educ.*, vol. 15, no. 3, pp. 292–297, 2017.

[11] H. Nurdiyanto, R. Rahim, and N. Wulan, "Symmetric Stream Cipher using Triple Transposition Key Method and Base64 Algorithm for Security Improvement," *J. Phys. Conf. Ser.*, vol. 930, no. 1, p. 12005, Dec. 2017.

[12] L. Yu, Z. Wang, and W. Wang, "The Application of Hybrid Encryption Algorithm in Software Security," *Fourth Int. Conf. Comput. Intell. Commun. Networks*, pp. 762–765, 2012.

[13] K. Fiscus and D. Shinburg, "Base64 Can Get You Pwned," 2011.

[14] G. Singh and Supriya, "Modified vigenere encryption algorithm and its hybrid implementation with Base64 and AES," in *Proceedings - 2nd International Conference on Advanced Computing, Networking and Security, ADCONS 2013*, 2013, pp. 232–237.