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/ijatcse126932020.pdf
https://doi.org/10.30534/ijatcse/2020/126932020

An Improved Affine Cipher using Blum Blum Shub Algorithm

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

Information and data security is the practice of digital privacy measures against illegal inspection, access, disclosure, use, recording, disruption, modification, or destruction of data regardless of the form the data may take. With this, cryptography, specifically the encryption and decryption process, has become an essential component of most data security strategies. In this paper, the encryption and decryption capability of the Affine cipher was investigated. The combination of Blum Blum Shub algorithm along with Affine cipher is proposed. In the proposed method, identical plaintext characters are not likely to be substituted with a similar sequence or pattern identical to the original plaintext. This is done by the use of the Blum Blum Shub algorithm. Simulation results revealed that the modified Affine cipher produces ciphertext that is more secure as it shows no trace of any pattern that is the same from the plain text.

Key words: Affine cipher, Blum Blum Shub, cryptography, encryption, hybrid ciphers

1. INTRODUCTION

Securing files has been essential, especially when data is very confidential [1]. Different cryptosystems are designed to encrypt files using different keys, retrieve original data properly via decryption of the ciphertext, encrypt and decrypt data messages accurately with higher security, and provide immunity against the access of unauthorized users [2].

The purpose of encryption is to make a file unusable if it is stolen, copied, downloaded, lost, or otherwise improperly accessed [2]. With this, data and information security are regarded as one of the most important aspects that everyone should consider not just only in business but in almost all sectors of society [3].

The introduction of encryption technology is the last line defense to ensure the security of information after an illegal retrieval of data is made, or the firewall and network protection have been broken in by unauthorized personnel. It helps to secure the content of the file by concealing and safely transforming text or into meaningless characters like a corrupted file, which will be avoided by the third party. In this paper, the Affine cipher is modified and is incorporated with Blum Blum Shub (BBS) algorithm to generate one of its secret keys. The proposed method randomizes the keys performing varied key shift and substitution that is used in the encryption and decryption of each character in the plaintext.

2. METHODOLOGY

2.1 Affine Cipher

In cryptography, the Affine cipher refers to a mono alphabetic substitution scheme based on the classical Caesar cipher. It is represented by the equation $A_{j,d}(x) = (jx + d) \bmod m$, where m is the range of alphabets, and j and d are secret keys [4]. Variables j and m should be coprime so that decryption can be made through the equation $A_{j,d}(y) \equiv j^{-1}(d - y) \bmod m$, where j^{-1} is the inverse modular multiplicative of modulo m that satisfies that equation $1 = aa^{-1} \bmod m$ [5]–[7].

Through modular arithmetic, the Affine cipher maps a set of alphabets to a range of integers that is, in turn, mapped to another set of alphabets using substitution to transform the plaintext into ciphertext [5], [6].

For example, encrypting the plaintext ASSESSED requires converting each character to its numerical equivalent referring to its index in the alphabet, such that A is 0 and Z is 25. The indexing of the Latin alphabets A to Z is presented in Table 1, while the index equivalents of the plaintext UNNEEDED represented as x is shown in Table 2.

Table 1: Alphabet indices

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Table 2: Plaintext numerical equivalent

Plaintext	A	S	S	E	S	S	E	D
x	0	18	18	4	18	18	4	3

With the affine encryption function $A_{j,d}(x) = (5x + 8) \bmod 26$, the plaintext ASSESSED is equivalent to 8 20 20 2 20 20 2 23 as presented in Table 3. These values are then converted to ciphertext using the affine table, as shown in Table 4.

Table 3: Encryption using Affine cipher

Plaintext	A	S	S	E	S	S	E	D
x	0	18	18	4	18	18	4	3
$(5x + 8) \bmod 26$	8	20	20	2	20	20	2	23
Ciphertext	I	U	U	C	U	U	C	X

Table 4: Affine table based on $A_{j,d}(x) = (5x + 8) \bmod 26$

Alphabet	Index	(5x+8) mod 26	Ciphertext
A	0	8	I
B	1	13	N
C	2	18	S
D	3	23	X
E	4	2	C
F	5	7	H
G	6	12	M
H	7	17	R
I	8	22	W
J	9	1	B
K	10	6	G
L	11	11	L
M	12	16	Q
N	13	21	V
O	14	0	A
P	15	5	F
Q	16	10	K
R	17	15	P
S	18	20	U
T	19	25	Z
U	20	4	E
V	21	9	J
W	22	14	O
X	23	19	T
Y	24	24	Y
Z	25	3	D

For decryption, the Affine cipher uses the equation $D(y) = 21(y - 8) \bmod 26$ where *21* is the modular multiplicative inverse a^{-1} of modulo 26, *y* is the numeric equivalent of the ciphertext character, and *8* is the number of shifts. For instance, the ciphertext IUUCUUCX is translated as 0 18 18 4 18 18 4 3 and decrypted as ASSESSED, as presented in Table 5.

Table 5: Decryption using Affine cipher

Ciphertext	I	U	U	C	U	U	C	X
y	8	20	20	2	20	20	2	23
21(y-8) mod 26	0	18	18	4	18	18	4	3
Plaintext	A	S	S	E	S	S	E	D

The Affine cipher produces obvious ciphertext patterns for plaintexts containing identical characters, just like any substitution ciphertext. Plaintext with repeating characters, such as S and E in Table 5, is also translated containing identical characters in the same sequence. Therefore, as soon as a certain repeating character is decrypted, the remaining identical characters can easily be identified as well, even without computation.

2.2 Blum Blum Shub

Blum Blum Shub (BBS) is a renowned probabilistically secure pseudo-random number generator (PRNG) proposed by Lenore Blum, Manuel Blum, and Michael Shub in 1986 [8]. The algorithm produces random numbers using two Blum-primes *p* and *q*, such that *p* and $q \equiv 3 \pmod 4$ and $gcd(p,q) = 1$. These primes, together with a random seed, is computed as:

$$b_{n+1} = b_n^2 \bmod M$$

where (i) *b* is a random seed value, (ii) *M* is the product of two large prime numbers *p* and *q, and* (iii) *p* and *q* are congruent to *3 (mod 4).*

The example below shows how BBS generates random numbers:

- Let $p = 7 \equiv 3 \pmod 4$;
  Let $q = 19 \equiv 3 \pmod 4$;
  Then, n = p * q = 7 * 19 = 133;
- Choose a seed value $b_0 = 100$
- $b_1 = 100^2 \pmod{133} = 25$
- $b_2 = 25^2 \pmod{133} = 93$
- $b_3 = 93^2 \pmod{133} = 4$
- *and so forth…*

2.3 Proposed Cipher Process

The proposed cipher process resolves the flaw of substitution ciphers by ensuring that distinct ciphertext values are produced, and no recognizable patterns appear for plaintext composed of identical characters such as AAAAAA or ABABABAB when encrypted. This is achieved by modifying the additive key in the affine cipher functions for every character encrypted. The proposed process introduces the use of the keystream randomly generated by the Blum Blum Shub algorithm. This stream of keys is relative to the size of the plaintext and is used as the value *d* in the encryption and decryption functions instead of a static digit.

The Affine cipher uses the equation $E(x) = (jx + y) \bmod 26$, where *j* must be coprime of *mod 26*; *x* is the character index, and *y* is a value from the unique keystream. Decryption is done using the equation $D(y) = j^{-1}(y - z) \bmod 26$, where j^{-1} is the modular multiplicative inverse of *mod 26; y* is the character index, and *z* is a value from the unique keystream. The encryption and decryption processes of the modified Affine cipher is shown in Figures 1-2.

To perform encryption using the modified Affine cipher, the following detailed processes are executed:

a. Identify the plaintext value. For example, the plaintext ASSESSED is used.
b. Identify the plaintext character index in the alphabet represented by *x,* as shown in Table 6.

Table 6: Plaintext character indices

Plaintext	A	S	S	E	S	S	E	D
Position *c*	1	2	3	4	5	6	7	8
Index *x*	0	18	18	4	18	18	4	3

c. Generate keystream values using the Blum Blum Shub algorithm. The keystream generated by BBS is 12 91 22 65 92 123 124 32. Each value is matched to each character in the plaintext, as shown in Table 7.

Table 7: Key stream value

Plaintext *p*	A	S	S	E	S	S	E	D
Position *c*	1	2	3	4	5	6	7	8
Index *x*	0	18	18	4	18	18	4	3
Key *y*	12	91	22	65	92	123	124	32

d. Generate ciphertext using the equation *E(x) = (5x + y) mod 26*. The plaintext ASSESSED with the keystream 12 91 22 65 92 123 124 32 is encrypted as MZIHAFOV, as presented in Table 8.

Table 8: Equivalent ciphertext

Plaintext	A	S	S	E	S	S	E	D
Position *c*	1	2	3	4	5	6	7	8
Index *x*	0	18	18	4	18	18	4	3
Key *y*	12	91	22	65	92	123	124	32
E(x) = (5x + y) mod 26	12	25	8	7	0	5	14	21
Ciphertext	M	Z	I	H	A	F	O	V

The following detailed processes are performed to decrypt using the proposed cipher:

a. Identify ciphertext value and keystream. For instance, the ciphertext is MZIHAFOV, and the unique keystream 12 91 22 65 92 123 124 32 are used.
b Identify plaintext character index in the alphabet represented by *x*.
c. Generate plaintext using the equation *D(y) = 21 (y - z) mod 26*. The ciphertext MZIHAFOV with the keystream 12 91 22 65 92 123 124 32 is decrypted as ASSESSED, as shown in Table 9.

Table 9: Equivalent plaintext

Ciphertext	M	Z	I	H	A	F	O	V
Index *y*	12	25	8	7	0	5	14	21
Key *z*	12	91	22	65	92	123	124	32
D(y) = 21 (y - z) mod 26	0	18	18	4	18	18	4	3
Plaintext	A	S	S	E	S	S	E	D

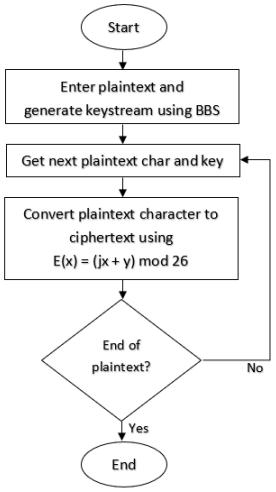

Figure 1: Modified affine cipher encryption process

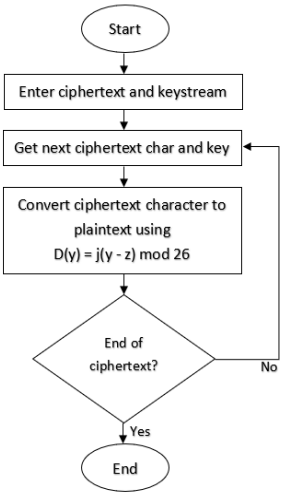

Figure 2: Modified affine cipher decryption process

3. RESULTS AND DISCUSSION

In order to assess the viability of the proposed method, it is tested using a variety of plaintext and keys, then compared versus the standard Affine cipher. The following test cases are shown in Tables 10-14.

Table 10: Test case 1

Plaintext	FOOLPROOF
Plaintext Size	9 bytes
BBS keystream	98 31 41 35 97 89 50 115 122
Standard Affine Ciphertext	EVVCCXCX
Modified Affine Ciphertext	TXHMQSQDR

Table 11: Test case 2

Plaintext	BBBBBBBBBBBBBBB
Plaintext Size	15 bytes
BBS keystream	70 3 0 79 26 41 43 76 1 41 55 60 100 113 16
Standard Affine Ciphertext	NNNNNNNNNNNNNN
Modified Affine Ciphertext	XIFGFUWDGUINBOV

Table 12: Test case 3

Plaintext	AABBAABBAABBAABBAABBAABB AABBAABBAABB
Plaintext Size	36 bytes
BBS keystream	112 39 42 22 119 20 29 113 37 72 124 107 55 32 108 113 7 49 83 17 40 71 100 91 113 122 67 73 60 25 38 103 33 57 118 19
Standard Affine Ciphertext	IINNIINNIINNIINNIINNIINNIINNIINN IINN
Modified Affine Ciphertext	INVBPUIOLUZIDGJOHXKWOTBSJS UAIZREHFTY

Table 13: Test case 4

Plaintext	CRYPTOGRAPHYISMAGIC
Plaintext Size	19 bytes
BBS keystream	114 30 76 30 75 56 76 20 28 48 35 86 9 2 101 4 23 77 31
Standard Affine Ciphertext	SPYFZAMPIFRYWUQIMWS
Modified Affine Ciphertext	ULOBOWCBCTSYXOFEBNP

Table 14: Test case 5

Plaintext	AFFINECIPHERENCRYPTIONAND DECRYPTIONPROCESS
Plaintext Size	42 bytes
BBS keystream	4 74 0 20 125 26 24 105 126 70 50 113 120 86 59 65 19 103 30 69 67 68 57 54 121 97 39 30 108 34 79 78 32 68 64 19 114 46 87 36 66 58
Standard Affine Ciphertext	IHHWVCSWFRCPCVSPYFZWAVIVX XCSPYFZWAVFPASCUU
Modified Affine Ciphertext	EVZIIUIPTBSQKVRUJWVFHDFPGI HOLYYRUIZQRMTEAS

Results show that the proposed process performs better in producing ciphertext, especially for plaintext values with identical characters. Characters A and B, as shown in Tables 11-12, are encrypted with obvious patterns using the standard affine method as opposed to the ciphertext generated by the proposed method. Cryptanalysis may be extensive and exhaustive since the modified Affine cipher uses dynamically generated keystream with the help of the Blum Blum Shub algorithm. These promising results make the proposed process more secure and are tough to crack in comparison to the standard Affine cipher.

4. CONCLUSION

In this paper, the additive key of the traditional Affine cipher that generates ciphertext based on the plaintext is modified. The incorporation of the Blum Blum Shub algorithm enables the generation of a random keystream to increase the unpredictability of ciphertext using the modified cipher. The proposed method produced a unique ciphertext through randomness, in which identical plaintext characters are not likely to be substituted with a similar sequence or pattern the same as the original plaintext. In general, the produced method produces a more secure information hiding process.

REFERENCES

[1] S. Soomro, M. R. Belgaum, Z. Alansari, and R. Jain, "Review and open issues of cryptographic algorithms in cyber security," in *International Conference on Computing, Electronics and Communications Engineering, iCCECE 2019*, 2019, pp. 158–162. https://doi.org/10.1109/iCCECE46942.2019.8941663

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

[3] A. Rayarapu, A. Saxena, N. V. Krishna, and D. Mundhra, "Securing Files Using AES Algorithm," *Int. J. Comput. Sci. Inf. Technol.*, vol. 4, no. 3, pp. 433–435, 2013.

[4] A. G. Konheim, *Computer Security and Cryptography*. Wiley, 2007. https://doi.org/10.1002/0470083980

[5] H. Zhu, C. Zhao, X. Zhang, and L. Yang, "An image encryption scheme using generalized Arnold map and affine cipher," *Opt. - Int. J. Light Electron Opt.*, vol. 125, pp. 6672–6677, 2014. https://doi.org/10.1016/j.ijleo.2014.06.149

[6] Y. S. Mezaal and S. F. Abdulkareem, "Affine Cipher Cryptanalysis Using Genetic Algorithms," *JP J. Algebr. Number Theory Appl.*, vol. 39, no. 5, pp. 785–802, 2017. https://doi.org/10.17654/NT039050785

[7] S. A. Babu, "Modification Affine Ciphers Algorithm for Cryptography Password," *Int. J. Res. Sci. Eng.*, vol. 3, no. 2, pp. 346–351, 2017.

[8] L. Blum, M. Shub, and M. Blum, "Simple Unpredictable Pseudo-Random Number Generator," *SIAM J. Comput.*, vol. 15, no. 2, pp. 364–383, 1986. https://doi.org/10.1137/0215025