Modified AES Algorithm Using Multiple S-Boxes

Felicisimo V. Wenceslao, Jr.
Graduate Programs
Technological Institute of the Philippines, Quezon City, Philippines
fvwenceslao@yahoo.com

Bobby D. Gerardo
Institute of Information and Communications Technology,
West Visayas State University,
Lapaz, Iloilo City, Philippines
bgerardo@wvsu.edu.ph

Bartolome T. Tanguilig III
College of Information Technology Education,
Technological Institute of the Philippines,
Quezon City, Philippines
bttanguilig_3@yahoo.com

ABSTRACT

This paper proposes for a modified version of the AES algorithm using multiple Sboxes. While many studies have been conducted specifically on modifying the Sbox, these studies were made to replace the Rijndael Sboxes in the AES cipher. We propose to implement two substitution boxes, where the first Sbox is the Rijndael Sbox and will be used as is. The second Sbox was constructed through an XOR operation and affine transformation and will replace the MixColumns operation within the internal rounds in the cipher. Based on simulation testing conducted, it was found out that the modified AES algorithm using multiple SBoxes has better speed performance compared to the original cipher. However, when tested using the avalanche effect, the changes in the output bits were below the minimum expected rate.

KEYWORDS

AES algorithm, Sbox**,** Cryptography, Affine Transformation

1 INTRODUCTION

In 1997, the National Institute of Standards and Technology (NIST) started a process to identify a replacement for the Data Encryption Standard (DES) which was generally recognized to be not secured due to fast advances in computer processing power. The goal of NIST was to define a replacement for DES that could be used for non-military information security applications by US government agencies. Additionally, commercial and other non-government users could also benefit from the technology as it can also generally adopted for commercial use.

The NIST invited experts in the field of cryptography and data security from around the world to participate in the discussion and in the selection process. There were five encryption algorithms that made to the final round of the screening process. Ultimately, the encryption algorithm proposed by the Belgium cryptographers Joan Daeman and Vincent Rijmen was selected. Prior to selection, Daeman and Rijmen used the name Rijndael (derived from their names) for the algorithm. After adoption, the encryption algorithm was given the name Advanced Encryption Standard (AES) which is in common use today[1].

In 2000, the NIST formally adopted the AES encryption algorithm and published it as a federal standard under the designation FIPS-197. It was chosen because of its security, performance, efficiency, implementability, and low memory requirements.

The MixColumn function in the AES algorithm is an important property of the cipher. Generally, it provides strength against differential and linear attacks due to the complexity of its mathematical operations. These complex mathematical operations may require computational resources in software implementation. We assume that by

replacing the MixColumn function, the speed performance of the AES algorithm will be improved.

In light of this study, this paper aims to propose for a modified AES algorithm using multiple S-Boxes. We will also compare the AES-Rijndael version and the modified AES algorithm using multiple S-Boxes and evaluate their performance and security properties.

2 REVIEW OF RELATED STUDIES

Modifying the AES cipher has been the subject of numerous studies. These range from changing the original Sbox using some other techniques. For instance, [2] proposed a substitution box that makes use of the RC4 key schedule algorithm (KSA). The resulting matrix is a key-dependent Sbox based that is dynamically generated based from some key. In their work, they constructed the Sbox-RC4 by:

- Running the RC4 KSA to construct 256! S-boxes depend on input key; and
- Perform an affine transformation to the RC4-KSA Sbox to produce the final Sbox.

The RC4-generated Sbox is use to replace the Rijndael Sbox during the encryption and decryption processes.

In [3], they proposed for yet another key-dependent Sbox that will substitute the Rijndael Sbox. In their paper, they modified the AES cipher by placing another phase in the beginning of the round function. They call the extra phase as the Sbox Rotation that will rearrange by way of rotating the Rijndael Sbox according to a round key. The round key is derived from the cipher key using the key schedule algorithm. The rotation value is dependent on the entire round key.

The results of their study showed that the enhancement on the original AES does not violate the security of the cipher. The enhanced version introduces confusion without violating the diffusion property.

In [4], proposed an enhanced version of the AES-128 algorithm by reducing the number of rounds from 10 rounds to 8 rounds. They assumed that with less number of rounds, it will result in less processing time of the AES algorithm. However, the reduction in the number of rounds to 8 is risky in security attacks such as differential attack and distinguishing attack. To offset such risk, their proposed enhanced AES-128 algorithm uses hashing function to compensate the attacks being mentioned. Hence, their enhanced AES algorithm, while less in the number of rounds yet an extra phase for the hashing function using the SHA-256 in every round is included. The result of their experiment revealed that the hashing function improved the security aspects of the cipher but required more number of operations.

3 BASIC CONCEPT OF THE AES ALGORITHM

The AES algorithm is a block cipher with a block length of 128 bits. The key which is provided as the input is expanded into an array of key schedule words, with each word has a size of four bytes. The total key schedule for the 128-bit key is 44 words.

AES allows for three different key lengths: 128, 192, or 256 bits. The encryption/decryption is consists of 10 rounds of processing for a 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. During the encryption and decryption processes, the 16 bytes of data will form a changeable (4*4) array called the state array[5].

For the encryption, the state array consists initially of the input data. This array will keep changing until the final encrypted data is reach. In the decryption process, the state array start from the enciphered data and will keep changing until the original data is produced. The encryption of AES is carried out in blocks with a

fixed block size of 128 bits each. The AES cipher calculation is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of cipher text. Figure 1 shows the AES cipher structure.

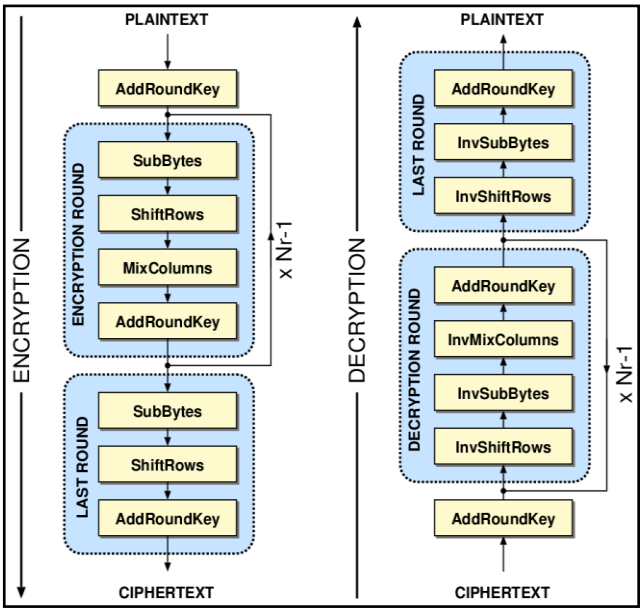

Figure 1. The AES Algorithm Structure.

Except for the last round in each case, all other rounds are identical. Inside each round are four different stages. These are:

3.1 Substitute Bytes (SubBytes Operation)

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box)[6]. This is a major reason for the security of the AES. With the help of this lookup table, the 16 bytes of the state (the input data) are substituted by the corresponding values found in the table**.** Figure 2 shows the SubBytes operation.

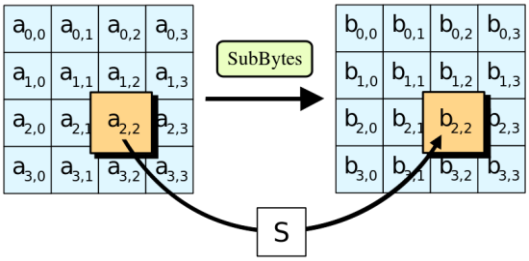

Figure 2. SubBytes Operation

3.2 Shift Rows (ShiftRows Operation)

The ShiftRows operation (figure 3) provides inter-column diffusion where the bytes in the last three rows of the states are shifted. Hence, the second row of the states is shifted by one byte position to the left of the matrix; the third row of the states is shifted by two bytes position to the left of the matrix; and the fourth row of the states is shifted by three bytes position to the left[6].

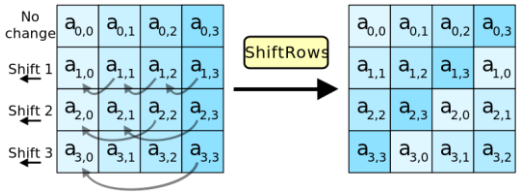

Figure 3. ShiftRows Operation

3.3 Mix Columns (MixColumns Operation)

Figure 4 shows the MixColumns operation that provides inter-byte diffusion where each column vector is multiplied by a fix matrix. The Galois Field is used in this operation. The bytes are treated as polynomials rather than numbers.

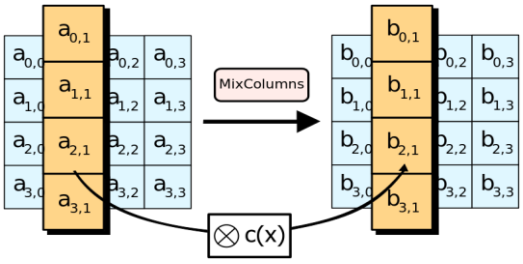

Figure 4. MixColumns Operation

3.4 Add Round Key (AddRoundKey Operation)

The AddRoundKey operation is simple. In this transformation, a round key is added to the state by a simple bitwise XOR operation. Each round key consists of *Nb* words from the key schedule[7]. It is performed by XOR-ing each byte of the state and the round key. Figure 5 shows the AddRoundKey operation.

ISBN: 978-1-9491968-07-9 ©2015 SDIWC

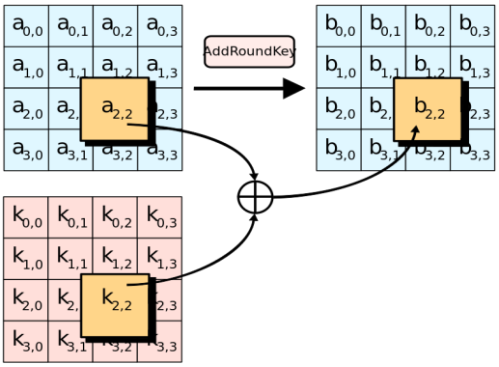

Figure 5. AddRoundKey Operation

For encryption, the individual transformations for the pseudocode computation consist of SubBytes(), ShiftRows(), MixColumns() and AddRoundKey(). These transformations play a role in processing the state[7]. The transformation (number of rounds) is performed depending on the key length. However, the final round is only consists of three stages: SubBytes, ShiftRows and AddRoundKey in producing the final encrypted data or ciphertext.

The decryption process is essentially the same structure as the encryption, following the nine rounds of Inverse ShiftRows, Inverse SubBytes, Inverse AddRoundKey and Inverse MixColumns transformation. In the final round, the Inverse MixColumns is no longer performed.

4 PROPOSED MODIFIED 128-AES ALGORITHM USING MULTIPLE S-BOXES

Among the four functions within rounds of the AES algorithm, the MixColumns function is perceive to be requiring more computational resources in software implementation as compared to the other functions. This is due to the fact that the MixColumns function provides the critical security properties of the cipher to avoid from linear and/or differential attacks. However, replacing the MixColumns function by an alternative process may increase the speed performance of the AES algorithm.

In this paper, we propose for a modified version of the 128-AES algorithm using two substitution boxes. The first S-Box is the Rijndael S-Box that is the default in the original structure of the cipher. The second S-Box is constructed using XOR operation and affine transformation. It will replace the MixColumns operations at each round as implemented in the original algorithm.

In essence, the encryption process of the modified 128-AES algorithm follows the sequence of SubBytes, ShiftRows, SubBytesXOR and AddRoundKey operations for nine rounds. In the final round, SubBytes, ShiftRows and the AddRoundKey operations will be performed to produce the ciphertext.

To decrypt, the modified 128-AES is perform using the sequence of Inverse ShiftRows, Inverse SubBytes, Inverse AddRoundKey and Inverse SubBytesXOR operations for nine rounds. In the final round, the Inverse SubBytesXOR is drop to produce the plaintext. Figure 6 shows the proposed modified AES algorithm structure using multiple S-Boxes.

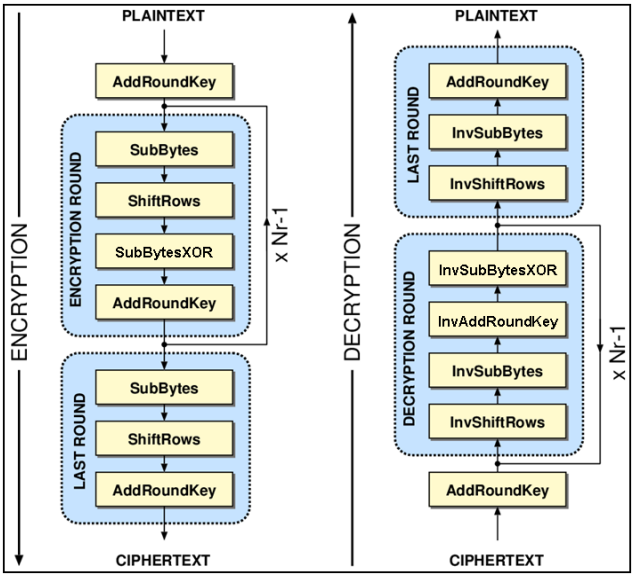

Figure 6. Modified AES Algorithm Using Multiple S-Boxes

5 CONSTRUCTION OF THE NEW S-BOX

The second S-Box is derived from the original S-Box as designed in the AES (hereafter referred as AES-Rijndael). It is constructed using the

following process:

5.1 Exclusive OR Operation

The first step is to do an XOR operation to the AES-Rijndael using some Key[i]. The Key[i] shall be any hexadecimal value between 00 to FF. In this particular matrix, the key used was 7F. Hence, the new S-Box shall be referred to as AES-2SboxXOR$_{7F}$. For the initial values of the AES-SboxXOR$_{7F}$, each cell in the AES-Rijndael will be XORed with 7F (AES-Rijndael[x,y] $\otimes$ 7F).

5.2 Affine Transform Operation

After creating the initial values of AES-2SboxXOR, each cell will be subjected to affine transformation, as applied to the Sbox-Rijndael, to avoid any fix points and to make the new S-box invertible.

To scramble the bits in each byte value, we next apply the following transformation to each bit b_i as stored in the initial AES-2SboxXOR$_{7F}$:

$$b'_i = b_i \otimes b_{(i+4) \bmod 8} \otimes b_{(i+5) \bmod 8} \otimes b_{(i+6) \bmod 8} \otimes b_{(i+7) \bmod 8} \otimes c_i \quad (1)$$

where c_i is the i^{th} bit of a specially designated byte c whose hex value is 0x63 (c7c6c5c4c3c2c1c0 = 01100011).

For the inverse AES-2SboxXOR, the following transformation to each bit was used for bit scrambling:

$$b'_i = b_{(i+2) \bmod 8} \otimes b_{(i+5) \bmod 8} \otimes b_{(i+7) \bmod 8} \otimes d_i \quad (2)$$

where d_i is the i^{th} bit of a specially designated byte d whose hex value is 0x05 (d7d6d5d4d3d2d1ddc0 = 00000101).

5.3 Matrix Mapping Operation

At the end of the affine transformation, the final values are known. The next step is to map each value to the matrix as appropriate to create the final lookup tables. Table 1 shows the final AES-2SboxXOR$_{7F}$ while table 2 shows the final Inverse AES-2SboxXOR$_{7F}$.

Table 1. AES-2SboxXOR$_{7F}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	13	0C	2D	32	6F	70	51	4E	EB	F4	D5	CA	97	88	A9	B6
1	E2	FD	DC	C3	9E	81	A0	BE	1A	05	24	3B	66	79	58	47
2	F0	EF	CE	D1	8C	93	B2	AD	08	17	36	29	74	6B	4A	55
3	01	1E	3F	20	7D	62	43	5C	F9	E6	C7	D8	85	9A	BB	A4
4	D4	CB	EA	F5	A8	B7	96	89	2C	33	12	0D	50	4F	6E	71
5	25	3A	1B	04	59	46	67	78	DD	C2	E3	FC	A1	BE	9F	80
6	37	28	09	16	4B	54	75	6A	CF	D0	F1	EE	B3	AC	8D	92
7	C6	D9	F8	E7	BA	A5	84	9B	3E	21	00	1F	42	5D	7C	63
8	9C	83	A2	BD	E0	FF	DE	C1	64	7B	5A	45	18	07	26	39
9	6D	72	53	4C	11	0E	2F	30	95	8A	AB	B4	E9	F6	D7	C8
A	7F	60	41	5E	03	1C	3D	22	87	98	B9	A6	FB	E4	C5	DA
B	8E	91	B0	AF	F2	ED	CC	D3	76	69	48	57	0A	15	34	2B
C	5B	44	65	7A	27	38	19	06	A3	BC	9D	82	DF	C0	E1	FE
D	AA	B5	94	8B	D6	C9	E8	F7	52	4D	6C	73	2E	31	10	0F
E	B8	A7	86	99	C4	DB	FA	E5	40	5F	7E	61	3C	23	02	1D
F	49	56	77	68	35	2A	0B	14	B1	AE	8F	90	CD	D2	F3	EC

Table 2. Inverse AES-2SboxXOR$_{7F}$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	7A	30	EE	A4	53	19	C7	8D	28	62	BC	F6	01	4B	95	DF
1	DE	94	4A	00	F7	BD	63	29	8C	C6	18	52	A5	EF	31	7B
2	33	79	A7	ED	1A	50	8E	C4	61	2B	F5	BF	48	02	DC	96
3	97	DD	03	49	BE	F4	2A	60	C5	8F	51	1B	EC	A6	78	32
4	E8	A2	7C	36	C1	8B	55	1F	BA	F0	2E	64	93	D9	07	4D
5	4C	06	D8	92	65	2F	F1	BB	1E	54	8A	C0	37	7D	A3	E9
6	A1	EB	35	7F	88	C2	1C	56	F3	B9	67	2D	DA	90	4E	04
7	05	4F	91	DB	2C	66	B8	F2	57	1D	C3	89	7E	34	EA	A0
8	5F	15	CB	81	76	3C	E2	A8	0D	47	99	D3	24	6E	B0	FA
9	FB	B1	6F	25	D2	98	46	0C	A9	E3	3D	77	80	CA	14	5E
A	16	5C	82	C8	3F	75	AB	E1	44	0E	D0	9A	6D	27	F9	B3
B	B2	F8	26	6C	9B	D1	0F	45	E0	AA	74	3E	C9	83	5D	17
C	CD	87	59	13	E4	AE	70	3A	9F	D5	0B	41	B6	FC	22	68
D	69	23	FD	B7	40	0A	D4	9E	3B	71	AF	E5	12	58	86	CC
E	84	CE	10	5A	AD	E7	39	73	D6	9C	42	08	FF	B5	6B	21
F	20	6A	B4	FE	09	43	9D	D7	72	38	E6	AC	5B	11	CF	85

6 EVALUATION RESULTS

6.1 Speed Performance

To test the speed performance of the proposed modified AES algorithm using multiple SBoxes, both versions were used to encrypt and decrypt a file with a size of 59 kilobytes for 20 trials.

For the encryption, the AES-Rijndael version obtained a mean of 171.75ms while the proposed AES-2SBox obtained a mean of 139.65ms. Using the Data Analysis Tool from the Microsoft Excel 2010™, the t-Test for independent samples was used to statistically compute the significant difference in the speed performance. Based from the result of the test, the obtained P-value was 5.33936E-07. This is lower than the 0.05 level of significance, hence there is indeed a significant difference in the speed performance favouring the proposed AES-2SBox version. Table 3 shows the t-Test Statistics for Independent Samples of the encryption process.

Table 3. Speed Performance Between the AES-Rijndael and AES-2SBox During Encryption

t-Test: Two-Sample Assuming Unequal Variances

	AES-Rijndael	*AES-2SBox*
Mean	171.75	139.65
Variance	359.1447368	190.7657895
Observations	20	20
Hypothesized Mean Difference	0	
Df	35	
t Stat	6.121727783	
P(T<=t) one-tail	2.66968E-07	
t Critical one-tail	1.689572458	
P(T<=t) two-tail	5.33936E-07	
t Critical two-tail	2.030107928	

For the decryption process, the AES-Rijndael obtained a mean of 195.00ms while the AES-2SBox version obtained a mean of 92.95ms. Similarly, the t-Test for Independent samples was used for statistical computation. Based from the result of the test, the P value obtained was 3.9442E-25 which is lower than the 0.05 level of significance. Hence, there is a significant difference in the speed performance with the proposed AES-2SBox performing better. Table 4 shows the t-Test statistics for independent samples of the decryption process.

Table 4. Speed Performance Between the AES-Rijndael and AES-2SBox During Encryption

t-Test: Two-Sample Assuming Unequal Variances

	AES-Rijndael	*AES-2SBox*
Mean	195	92.95
Variance	168.6315789	167.5236842
Observations	20	20
Hypothesized Mean Difference	0	
Df	38	
t Stat	24.89190009	
P(T<=t) one-tail	1.97221E-25	
t Critical one-tail	1.68595446	
P(T<=t) two-tail	3.94442E-25	
t Critical two-tail	2.024394164	

6.2 Test for Avalanche Effect

The avalanche effect refers to a desirable property of cryptographic algorithms. The avalanche effect is evident if, when an input is changed slightly (for example, flipping a single bit) the output changes significantly with at least half the output bits will be flip[8].

In[9], the calculation of avalanche effect can be derived by using equation:

$$\text{Avalanche Effect (\%)} = (NC/TN) * 100 \quad (3)$$

where NC is the number of changed bits in ciphertext and TN is the total number of bits in the ciphertext.

Here, we start to calculate the avalanche effect of the AES-2Sbox. The tests were performed by changing the plaintext bit from “11” to “10” and from “FF” to “F0”. The results obtained were 24.219% with 31 bits that were changed and 19.531% with a flip of 25 bits respectively. The following table shows the result of the test for avalanche effect for AES-2Sbox.

Table 4. Avalanche Effect of the AES-2SBox

Plaintext	Ciphertext	Avalanche Effect
11111111111111111111111111111111	FE88B9C6D624C203A4345796445320E4	24.219% (31)
11111111111111111111111111111110	40A3CFC6D624C2D972345796B15320A4	
00112233445566778899AABBCCDDEEFF	9E53C352DBDF8A4F1034CABEAC05B1FB	19.531% (25)
00112233445566778899AABBCCDDEEF0	37F1B112DBDF8A221034848DAC05B1FB	

7 CONCLUSION

This paper presents a proposed modified AES algorithm using multiple Sboxes. The first Sbox (AES-Rijndael) stand as is in the cipher structure. Meanwhile, a new Sbox was constructed using XOR operation and affine transformation. This Sbox, which we call AES-2SBoxXOR, replaced the MixColumns step in the AES cipher rounds.

Two set of tests were conducted to the original version and the modified version of the AES algorithm. In the speed performance test, where the two versions, through an implementation model, encrypted and decrypted a file with a size of 59Kb for 20 trials. The t-Test statistics set at 0.05 level of significance revealed that in both encryption and decryption, there is a significant difference in the speed performance favouring the proposed AES-2SBox version with the obtained P-values 5.33936E-07 and 3.94442E-25 respectively.

We also performed the test of avalanche effect on the proposed AES-2SBox algorithm. The results of the simulation revealed that the avalanche effect is slightly lower than the minimum expected output of at least 50% bit flip when 1 bit input is altered. The obtained changes in the bit sequence were computed at 24.219% and 19.531% for two set of plaintext.

From these results, we observed that the speed performance greatly increased in the modified AES algorithm using multiple S-Boxes, while the security side has slightly weakened.

REFERENCES

[1] Townsend Security, Introduction to AES Encryption: White Paper, TownSendSecurity.com, DOI= https://townsendsecurity.com/sites/default/files/AES_Introduction.pdf, 2010.

[2] I. Abd-ElGhafar, A. Rohiem, A. Diaa and F. Mohammed, Generation of AES Key Dependent S-Boxes using RC4 Algorithm, [13 International Conference on Aerospace Sciences & Aviation Technology (ASAT- 13). May 26 – 28, 2009, Military Technical College, Kobry Elkobbah, Cairo, Egypt, 2009].

[3] J. Juremi, R. Mahmod, S. Sulaiman and J. Ramli, Enhancing Advanced Encryption Standard S-Box Generation Based on Round Key, [International Journal of Cyber-Security and Digital Forensics (IJCSDF) 1(3): The Society of Digital Information and Wireless Communications (SDIWC) 2012 (ISSN: 2305-0012), pp. 183-189, 2012].

[4] S. Manasa, P. Mullaimalar, G. B. Gnanaprakash Singh, and Prof. S. S. Manivannan, Reducing the Key Generation Time Using Enhanced AES-128 Algorithm to Secure the Data over Wireless Networks, [International Journal of Applied Engineering Research, ISSN 0973-4562, Vol. 8, No. 19, pp. 2453-2456, 2013].

[5] V. Pachori, G. Ansari, and N. Chaudhary, Improved Performance of Advance Encryption Standard using Parallel Computing, [International Journal of Engineering Research and Applications. Vol. 2, Issue 1, Jan-Feb 2012, pp. 967-971, 2012].

[6] Federal Information Processing Standards Publication 197, Announcing the Advanced Encryption Standard, DOI = http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 2001.

[7] Man Young Rhee, Internet Security: Cryptographic Principles, Algorithms and Protocols. John Wiley & Sons Ltd: England, 2003, p.114.

[8] WikiPedia.Com, Avalanche Effect, DOI = http://en.wikipedia.org/wiki/Avalanche_effect, 2014.

[9] G. Patidar, N. Agrawal and S. Tarmakar, A block based Encryption Model to improve Avalanche Effect for Data Security, [International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013].

ABOUT THE AUTHORS

Felicisimo V. Wenceslao, Jr. finished his Bachelor of Science in Computer Science at the Computer College of the Visayas, Iloilo City, Philippines (1993) and his Master of Science in Information Technology at Hannam University, Republic of Korea (2005). He is currently pursuing his Doctor in Information Technology at the Technological Institute of the Philippines, Quezon City, Philippines. He is currently the Director of the Institute of Information and Computer Studies at Northern Iloilo Polytechnic State College, Estancia, Iloilo, Philippines. His research interests are in network security, mobile development, data mining and e-learning.

Dr. Bobby D. Gerardo is currently the Vice President for Administration and Finance of West Visayas State University, Iloilo City, Philippines. His dissertation: Discovering Driving Patterns using Rule-based intelligent Data Mining Agent (RiDAMA) in Distributed Insurance Telematic Systems. He has published more than 60 research papers in national and international journals and conferences. He is a referee to international conferences and journal publications such as in IEEE Transactions on Pattern Analysis and Machine Intelligence and IEEE Transactions on Knowledge and Data Engineering. He is interested in the following research fields: distributed systems, telematics systems, CORBA, data mining, web services, ubiquitos computing and mobile communications.

Dr. Bartolome T. Tanguilig III took his Bachelor of Science in Computer Engineering in Pamantasan ng Lungsod ng Maynila, Philippines in 1991. He finished his Master's Degree in Computer Science from De La Salle University, Manila, Philippines in 1999, and his Doctor of Philosophy in Technology Management from Technological University of the Philippines, Manila in 2003. He is currently the Assistant Vice President for Academic Affairs and concurrent Dean of the College of Information Technology Education and Graduate Programs of the Technological Institute of the Philippines, Quezon City.

Dr. Tanguilig is a member of the Commission on Higher Education (CHED) Technical Panel for IT Education, the chair of the CHED Technical Committee for IT, the founder of Junior Philippine ITE Researchers (JUPITER), board member of the Philippine Society of IT Educators (PSITE), member of the PCS Information and Computing Accreditation Board (PICAB), and a member of the Computing Society of the Philippines (CSP).