A Multi-Threaded Symmetric Block Encryption Scheme Implementing PRNG for DES and AES Systems

Adi A. Maaita
Department of Software Engineering
Faculty of Information Technology, Isra University
Amman, Jordan

Hamza A. Alsewadi
Faculty of Information Technology
Middle East University
Amman, Jordan

***Abstract*—Due to the ever-increasing efficiency of computer systems, symmetric cryptosystem are becoming more vulnerable to linear cryptanalysis brute force attacks. For example, DES with its short key (56 bits) is becoming easier to break, while AES has a much longer key size (up to 256 bits), which makes it very difficult to crack using even the most advanced dedicated cryptanalysis computers. However, more complex algorithms, which exhibit better confusion and diffusion characteristics, are always required. Such algorithms must have stronger resistance against differential and linear cryptanalysis attacks. This paper describes the development of an algorithm that implements a pseudo random number generator (PRNG) in order to increase the key generation complexity. Experimental results on both DES and AES cryptosystems complemented with the PRNG have shown an average improvement of up to 36.3% in the avalanche error computation over the original standard systems, which is a considerable improvement in the time complexity of both systems.**

Keywords—Computer Security; Symmetric cryptography; DES; AES; pseudo random number generators

I. Introduction

Governments, banks, universities, and regular individuals are sending and receiving colossal amounts of digital data over networks and through other digital means non-stop. The ever flowing torrent of data holds information of varying levels of importance and sensitivity, such of which is determined by the purpose to which it will be put to use by its sender and receiver, and the damage which results from it falling into the wrong hands.

Keeping government, industrial, financial, and personal secrets safe is a paramount concern in a world controlled through digital communications and integrated data storage. Secrets flow from one computer to another until they reach their designated destinations. But what if those secrets were intercepted?

Encryption is an ancient solution designed to protect information which can be intercepted by those who were not meant to receive it. Many algorithms were developed over thousands of years for that purpose. In the digital age, encryption algorithms are classified into symmetric algorithms (secret-key algorithms), and asymmetric algorithms (public-key algorithms) [1, 2]. Symmetric algorithms require both the sender and the receiver of encrypted data to have the same key which will be used for both encryption and decryption, while for asymmetric algorithms, the key used to perform encryption of some data is different from the key which will be used to decrypt that data.

This work is concerned with the enhancement of the secret key generation process using random number generator, that will be used with symmetric cryptographic systems, in particular data encryption standard (DES) and advanced encryption standard (AES). Hence only these two systems will be reviewed together with pseudo random number generators in the following sections. After the brief introduction and the literature review presented in sections 1 and 2, section 3 will include the methodology of the proposed algorithm. Section 4 lists out the obtained results. Section 5 provides a comprehensive discussion of the obtained results. Finally, section 6 concludes the work.

II. Literature Review

This paper is concerned with the improvement of two widely used symmetric cryptosystems: the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES), by the implementation of a pseudorandom number generator (PRNG). Hence, a brief literature review will be included in this section.

A. The Data Encryption Standard (DES)

The widely used DES crypto-system was first developed by an IBM team and modified by the National Security Agency (NSA) to be adopted by the National Bureau of Standards (NBS) in 1976. It is an iterative block cipher system with a block size of 64 bits. It implements 16 rounds using a 56-bit key that changes for each round according a key generation algorithm. Confusion and diffusion were guaranteed through various substitution and transposition steps [1-4]. It was standardized in 1977 by the National Institute of Standards and Technology, and used internationally since then. It was secure enough at the beginning, however, due to its comparatively small key space, the existence of some weak and semi-weak keys, and the vast increase in the computing power, breaching its security became an easy task.

The DES algorithm weakness and vulnerability was exploited in the last decade of the twentieth century. Electronic Frontier Foundation was able to break DES in 1998 using the so called DES cracker [5]. Around the same period, DESCHEALL project, led by Rocke Verser, Matt Curtin, and Justin Dolske, were also able to break DES. They used idle cycles of thousands of computers across the Internet.

Encryption twice using DES (or 2DES) which doubles the key length to 112 bit was suggested as a modification to DES, but unfortunately, it suffered from man-in-the-middle attack. This drawback lead to a minor improvement in the key space by only increasing the key length from 56 to 57 bits [6].

The drawbacks of 2DES lead to the development of 3DESs which was a far more secure cryptosystem than DES. It was developed by an IBM team in 1999. The application of 3DES with three different keys extends the key space by practically achieving a key length of 168-bit, thus securing the system for few more years to come [7].

Many other variants of DES with less computational efforts were suggested, such as DES-X with key space enhanced by XOR'ing with other elements before and after the encryption process, and GDES that speeds up encryption. However, they were susceptible to differential cryptanalysis [8, 9].

The need arose for a successor to DES, and accordingly, National Institute of Standards and Technology (NIST) put forward a competition for designing a strong encryption algorithm. The criteria for the competing algorithms were to be efficient and easy to implement using both hardware and software, besides being royalty-free in order to be used internationally [10]. This competition was won by the Belgian cryptographers Joan Daemen and Vincent Rijmen in 2000 [11] and was named as the Rijndael algorithm, carrying the acronym of some characters of their names (pronounced "Rhine doll"). Then this algorithm wass termed Advanced Encryption Standard (AES) and defined by FIPS 197. It was approved by the US government to be used for secret and top Secret classified information [12].

B. The Advanced Encryption Standard (AES)

AES is an iterative symmetric cryptosystem operating on 128 bits data block size, i.e. double the data size for DES. There are three variants of AES according to the key lengths; 128, 192 and 256 bits, and the number of rounds; 10, 12, and 14 rounds. These increases in block size, key length and the number of rounds have given the AES algorithm dramatic security improvements as compared to DES when a brute force attack is used, besides no trace of "weak" and "semi-weak" keys are detected so far.

Diffusion and confusion are achieved in the AES computation through four operations that are executed in every round. Those operations are: byte substitution, shift rows, mix columns and add round keys. Also, an excellent key generation algorithm is implemented to produce a different key for each round. It implements S-box tables resulting from transformation using the Galois Field GF(2^8). It defines the transformation algebraically using the GF(2^8) field with the irreducible polynomials ($x^8 + x^4 + x^3 + x + 1$) [9, 10]. The detailed design and operation of the AES algorithm will not be listed here but can be found in the literature [9–12].

Although potential attacks against the AES algorithm, such as interpolation, saturation, Gilbert-Minier, truncated differential, and related-key attacks were suggested by Rijndael, most attacks have focused on the "side-channels", which rely on weaknesses in the security of the application rather than the algorithm [11]. Besides the strength of its security, AES can efficiently be implemented in both hardware and software, which makes it safe and practically beneficial now and for years to come.

C. Pseudo Random Number Generators (PRNGs)

Deterministic or Pseudo-random number generators are algorithms used to generate sequences of numbers having an approximate random property [13]. Pseudo-random number generation is initiated using relatively small key seeds, and the numbers are easy to generate and reproduce. PRNGs are classified into integer generators, sequence generators, integer set generators, narrators, sequence generators, integer set generators, Gaussian generators, decimal fraction generators or row random byte generators. This classification is based on the type of data they produce, such as integers, integer sequences, sets of random integers, integers that fit normal distribution or numbers in the 0 and 1 range with configurable decimal places, respectively. Each of the mentioned types is useful for many cryptographic purposes [14].

Some PRNGs generate pseudo-random numbers using seeds supplied by chaotic systems (dynamic, iterative, decimation) to achieve high speed and good security [15-17]. They are advantageous in having unpredictability or disorder-like, that are required for generating complex sequences. However, they have the problems of non-ideal distribution and short cycle length.

Behnia et. al. proposed a cryptographically secure algorithm for the generation of PRNGs based on three coupled and mutually perturbed Lagged Fibonacci generators [18]. It includes bitwise XOR cross-addition of each generator output with the right-shifted output of the nearby generator. It showed enhanced entropy and acceptable repetition period than the conventional Lagged Fibonacci Generator.

An enhancement to the work discussed above was done through a multi-stage PRNG algorithm that is based on Shannon's concept of confusion and diffusion. This algorithm was designed and tested for randomness using NIST randomness tests by the authors [19, 20]. It implements bitwise manipulation in order to achieve adequate bit string confusion and diffusion by combining various processes such as bit swapping, modular operations and secret splitting techniques. This algorithm will be implemented in this work to improve the key generation and manipulation of symmetric cryptosystems, such as DES and AES.

III. The Multi-Threaded Block Encryption Scheme

This paper proposes a multi-threaded block encryption scheme (MTBES). It is designed with the notion to enhance symmetric cryptosystems by improving the diffusion and confusion processes. This will be done through the introduction of more randomness into the key generation algorithms and utilizing the multi-threaded features in modern computers. In this work, the two widely used cryptosystems, namely DES and AES will considered. Each of these systems includes a number of rounds, where each round requires a certain sub-key. These sub-keys are normally generated by an algorithm that starts with an input secret key. Basically, this research work suggests two modifications; first, the incorporation of a pseudo-random number generator that participates in the generation of the rounds sub-keys needed for either DES or AES cryptosystems. Second, splitting the original message into sets of packets through various threads in the processer, that will be encrypted concurrently, each thread uses the suitable PRNG sub-keys, and then in the end, they are mixed and transmitted to the recipient where the packets are sorted and then decrypted. These two modifications are described in the following sections.

A. Sub-key Generation

Generally, each cryptosystem requires a secret key of certain length, namely it is of 64 bits length for DES and 128, 192, or 256 bits for AES. This key is normally used to generate a set of sub-keys $K = \{K_1, K_2, \ldots, K_n\}$ according to fixed procedure, where n is the number sub-keys required by the system. The number of sub-keys depends on the system used, namely 16 sub-keys for DES and 10, 12, or 14 sub-keys for AES different key length 128, 192 0r 256 bits, respectively (as shown in fig 1 for DES for example).

In this paper, a PRNG is used to randomly generate another set of n sub-keys, $S = \{S_1, S_2, \ldots, S_n\}$. To generate this set of sub-keys, PRNG requires a secret key, too to be used as seed. Each of these sub-keys length is the same as that for K and S. Next, the generated sub-keys, K and S are XOR'ed with each other producing a set of sub-keys K_i as illustrated in Figure I.

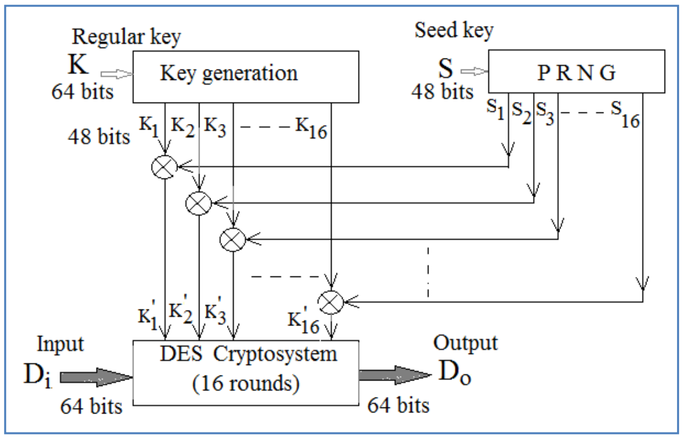

Fig. 1. Block diagram for the proposed sub-key generation scheme

This resulting set of sub-keys will be the one used for the successive rounds of the system under consideration.

As an example, the PRNG implemented in this work that combines logical operation and bits manipulation to achieve the confusion and diffusion concept. It accepts a certain secret key (as a seed) of the required length consisting of any alphanumeric and special characters agreed upon by the communicating parties. The produced sub-keys lengths and the secret seed length depend on the cryptosystem under consideration, (for example, 48 bits for DES and 128, 196, or 256 bits depending on the AES type used). This PRNG is designed and tested for accepted randomness using NIST randomness criterion [20].

B. Multi-threaded Operation

A program is written to arrange the algorithm execution through a multi-threaded processor, which means that its operation is divided over a number of threads. The number of threads is determined by the size of the data to be encrypted, as each thread should be responsible for encrypting a piece of the original text. The number of threads is determined by reading the threading capability of the CPU from the OS and segment the data to fit such threading capability. This process has exhibited an efficient execution practice that is expected to enhance the time complexity measurement.

Multi-threaded programming is used to enhance the performance of the algorithm when it is executed on a computer supporting multi-threading. However, the algorithm operates perfectly on a single core processor that does not support multi-threading.

The algorithm utilizes multi-threading by splitting the data to be encrypted into a number of packet lists equal to the possible number of threads supported by the processor. Each packet of each packet list is then encrypted using a separate thread, and then added to a master packet array in their original order. This order is preserved regardless of unpredictability of thread execution behavior as a packet is placed in the correct location within the array. This master array of packets is then converted to a string representing the final encrypted message, which is finally sent to the recipient.

IV. Experimental Results

The proposed algorithm is incorporated in both DES and AES cryptosystems in order to change them to modified versions, named randomized key DES (named RKDES) and randomized key AES (named RKAES).

The criteria used for the test is the average avalanche effect. The avalanche phenomena may be defined as the percentage of change in the ciphertext contents when the input plaintext is altered. The resulting Average Avalanche Effect percentage (AAE) for these algorithms are compared with original DES and AES cryptosystems running on the same computing environment. Moreover, different input plaintext lengths were considered ranging from 512 bits to 1048576 bits with various number of iterations ranging from 100 to 10000 epochs. These experiments were repeated for three different combinations of input data, namely, numeric only, alphanumeric and Unicode. In the following, some selected results are displayed.

The average avalanche effect (AAE) percentage for the original AES and the modified AES with random key RKAES are calculated for different data sizes ranging from 512 to 1048576 bits, and for different numbers of iterations ranging from 100 to 10000 iterations. The obtained results for the case of 10000 iterations are listed in tables I, II, and III. A graphical representation of the data is shown in the figures II, III, and IV.

TABLE I. AVERAGE AVALANCHE EFFECT OF AES AND RKAES FOR NUMERIC DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of AES and RKAES for Numeric data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	36.50%	47.30%
4096	37.00%	47.60%
65536	37.10%	47.90%
1048576	37.60%	52.90%
Average	**37.05%**	**48.93%**

TABLE II. AVERAGE AVALANCHE EFFECT OF AES AND RKAES FOR ALPHANUMERIC DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of AES and RKAES for Alpha-numeric data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	36.80%	47.60%
4096	36.80%	47.70%
65536	36.90%	47.90%
1048576	36.80%	53.80%
Average	**36.83%**	**49.25%**

TABLE III. AVERAGE AVALANCHE EFFECT OF AES AND RKAES FOR UNICODE DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of AES and RKAES for Unicode data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	36.50%	48.40%
4096	36.70%	48.50%
65536	36.80%	48.80%
1048576	36.70%	54.20%
Average	**36.68%**	**49.98%**

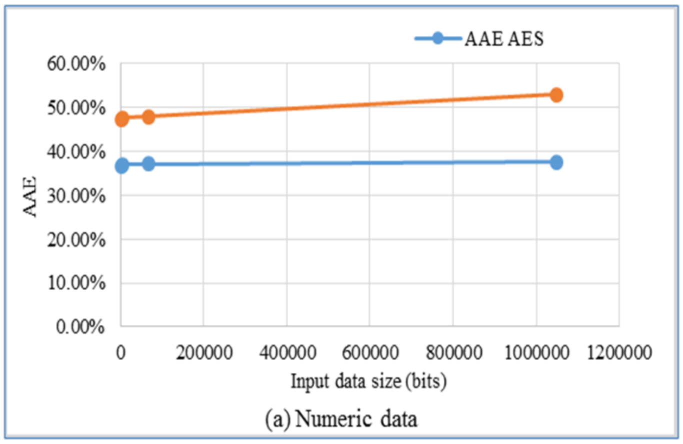

Fig. 2. Average Avalanche Effect of AES and RKAES for Numeric data after 10000 iterations

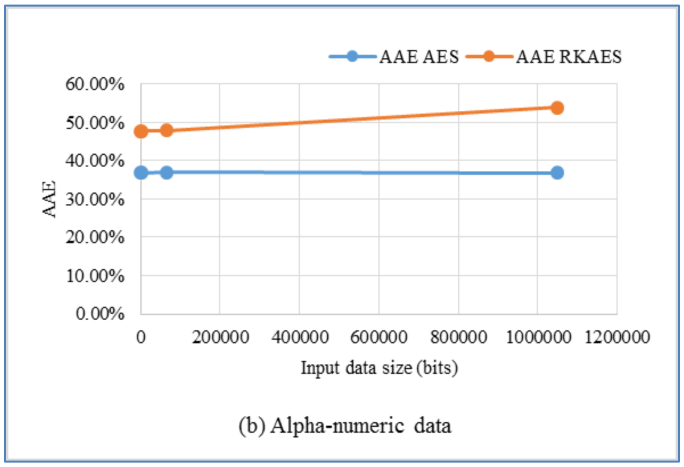

Fig. 3. Average Avalanche Effect of AES and RKAES for Alphanumeric data after 10000 iterations

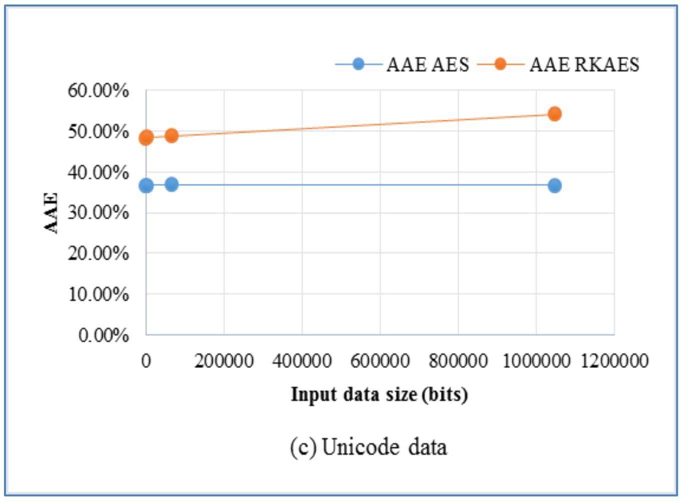

Fig. 4. Average Avalanche Effect of AES and RKAES for Unicode data after 10000 iterations

Similarly, the average avalanche effect (AAE) percentage for the original DES and the modified DES with random key RKDES are calculated for different data sizes ranging and for different numbers of iterations as those for the AES cryptosystem and the obtained results for the case of 10000 iterations are listed in tables IV, V, and VI, and illustrated in the figures V, VI, and VII for the three types of data.

TABLE IV. AVERAGE AVALANCHE EFFECT OF DES AND RKDES FOR NUMERIC DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of DES and RKDES for Numeric data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	23.90%	37.50%
4096	23.80%	37.40%
65536	23.70%	37.60%
1048576	23.80%	38.10%
Average	**23.80%**	**37.65%**

TABLE V. AVERAGE AVALANCHE EFFECT OF DES AND RKDES FOR ALPHANUMERIC DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of DES and RKDES for Alpha-numeric data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	24.20%	38.00%
4096	24.40%	38.10%
65536	24.70%	38.70%
1048576	24.80%	39.40%
Average	**24.53%**	**38.55%**

TABLE VI. AVERAGE AVALANCHE EFFECT OF DES AND RKDES FOR UNICODE DATA AFTER 10000 ITERATIONS

Average Avalanche Effect of DES and RKDES for Unicode data after 10000 iterations		
Data size	*AAE AES*	*AAE RKAES*
512	24.50%	39.00%
4096	24.40%	39.00%
65536	24.90%	39.70%
1048576	25.10%	40.60%
Average	**24.73%**	**39.58%**

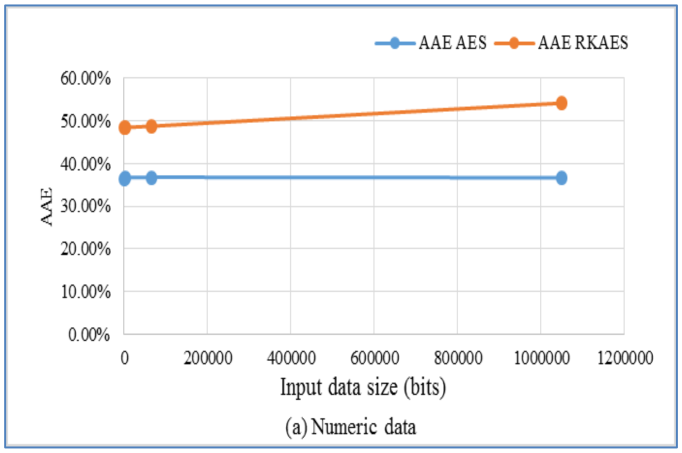

Fig. 5. Average Avalanche Effect of DES and RKDES for Numeric data after 10000 iterations

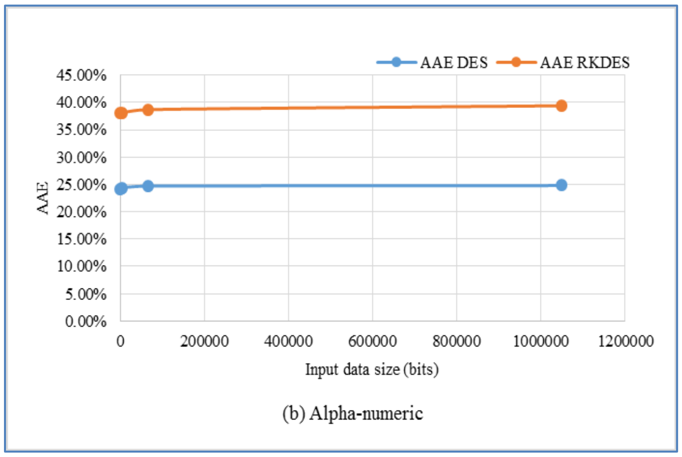

Fig. 6. Average Avalan*che Effect of DES and RKDES for* Alphanumeric data after 10000 iterations

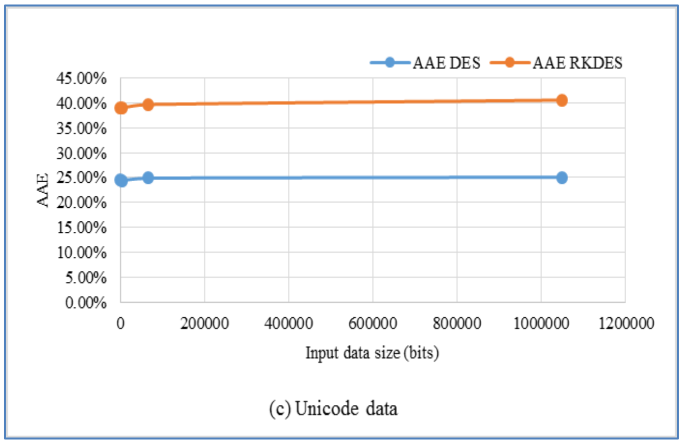

Fig. 7. Average Avalanche Effect of DES and RKDES for Unicode data after 10000 iterations

The average improvement of the avalanche effect when the key is randomized by the incorporation of the PRNG in the sub-key generation can be calculated by the formula shown in equation (1).

Average improvement of the avalanche effect,

$$\zeta = \frac{\Delta\ AAE}{AAE}\ \% \qquad . \quad . \quad . \quad . \qquad (1)$$

Computing the average improvement of the avalanche effect, ζ for various combinations of data types, input sizes, and number of iterations performed produced the results listed in table III.

TABLE VII. AVERAGE IMPROVEMENT OF THE AVALANCHE EFFECT ζ

Data type	No. of iterations	Average Improvement, ζ (%)	
		DES	AES
Numeric	100	32.53	32.7
	1000	32.05	32
	10000	32.06	32.1
Alphanumeric	100	33.19	33.2
	1000	34.16	34.2
	10000	33.75	33.7
Unicode	100	32.22	32.2
	1000	35.29	35.3
	10000	36.26	36.3
Average Improvement, ζ		33.50	33.52

Table III indicates a considerable improvement in cryptographic strength or system security. The calculated average avalanche effect showed and improvement of more than 33%. Actually the average improvement when various parameters are considered for DES was 33.50% and for AES was 33.52 %, which are almost equal. Such improvement has resulted from the involvement of the PRNG involvement in generating the sub-keys, which indicates that such technique would prove useful in other block cipher systems.

V. RESULTS ANALYSIS

Application of the proposed PRNG algorithm modification as part of the sub-key generation process within AES and DES algorithms has resulted into considerable improvements in the diffusion attribute of both algorithms. This was observed through the considerable increase in the avalanche effect (AAF), which was measured using a custom software package, developed for the testing of encryption strength attributes of block ciphers. The avalanche effect measurements for DES showed an increase by 33.5% in the case of the modified algorithm compared to the original DES, while that for AES showed an enhancement of 33.52% in the avalanche effect in the case of the modified algorithm compared to the original AES.

It can also be stated that the incorporation of PRNG as part of the sub-key generation process, can be considered a form of cryptography applied on the original key and subsequent sub-keys, in a cascading manner. This leads to what is known as domino effect that enhances the confusion and diffusion attributes for block ciphers by applying a multi-stage sub-key generation process.

Moreover, the incorporation of a key encryption algorithm exhibiting highly random outcomes, as measured by the NIST pseudo-randomness tests, such as the implemented PRNG in this work, would lead to enhanced bit diffusion behavior within multi-stage, multi-sub-key block ciphers such as DES and AES which were considered here.

From tables VII, it can be observed that the average avalanche effect for data constructed from larger alphabets was greater than that observed for data constructed from smaller alphabets. Namely, data in Unicode format showed a larger enhancement in the average avalanche effect than alpha-numeric data, and numeric data for the same data size and number of iterations involved. This is also manifested when comparing the average avalanche effects for numeric and alpha-numeric data, i.e. alpha-numeric data, shows better enhancement than numeric data. Besides, when different data sizes are compared, larger data samples showed better enhancement in the average avalanche effect than smaller data samples.

VI. CONCLUSIONS

Significant improvement has clearly resulted due to the incorporation of pseudo-random number generation into the sub-key generation process for both DES and AES algorithms. This means that such a process significantly enhances the diffusion property of the algorithm. This in turn has led to a higher level of security than those obtained using the original algorithms. Moreover, it was noticed that the average avalanche effects get better as one goes from numeric to Unicode through alphanumeric data with increasing number of iterations.

Furthermore, security is achieved by splitting the original message into packets, where each set of packets is encrypted using a pseudo-random sub-key. Using different sub-keys for encrypting sets of packets increases the difficulty of cryptanalysis through differential attacks which require the presence of a large number of original messages and their corresponding cipher texts.

REFERENCES

[1] Bruce Schneier, 1996, “Applied Cryptography: protocols, algorithms and source code in C”, John Wiley & Sons.

[2] William Stallings & Lawrie Brown, 2015, “Computer Security: Principles and Practice”. 3rd Ed., Pearson Press.

[3] M. Ebrahim , S. Khan, and U. Bin Khalid, “Symmetric Algorithm Survey: A Comparative Analysis”, International Journal of Computer Applications, Vol. 61, No.20, January 2013

[4] FIBS, FIPS PUB 46-3 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[5] Curtin, M and J. Dolske, “A Brute-Force Search of DES Keyspace”, Login: The Usenix Magazine, Vol. 23, No. 3, May 1998.

[6] Andrew D., K., “Computer Security”, Michaelmas, Oxford University, 2014. http://www.cs.ox.ac.uk/andrew.ker/docs/computersecurity-lecture-notes-mt2014.pdf.

[7] Noura Aleisa, “A Comparison of the 3DES and AES Encryption Standards”, International Journal of Security and Its Applications Vol.9, No.7, 2015, pp.241-246.

[8] Eli Biham and Adi Shamir, “Differential Cryptanalysis of the Data Encryption Standard”, Springer New York, Nov 9, 2011.

[9] William Stallings, “Cryptography and Network Security: Principles and Practice”, Pearson Education, Prentice Hall, Feb 18, 2016.

[10] Behrouz Forouzan, “ Cryptography and Network Security”, McGraw-Hill, 2008.

[11] Joan Daernen · Vincent Rijrnen, “The Design of Rijndael AES-The Advanced Encryption Standard” https://autonome-antifa.org/IMG/pdf/Rijndael.pdf

[12] Federal Information Processing Standards Publication 197, “Announcing the Advanced Encryption Standard (AES), 2001.http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[13] D. Dilli, and S. Madhu, “Design of a New CryptographyAlgorithm using Reseeding -Mixing Pseudo Random Number Generator,” IJITEE, vol. 52, no. 5, 2013.

[14] J. M. Bahi, and C. Guyeux, “Topological chaos and chaotic iterations, application to hash functions,” IEEE World Congress on Computational Intelligence WCCI’, Barcelona, Spain, July 2010. Best paper award, PP 1–7,

[15] J. Bahi, C. Guyeux, and Q. Wang, “A novel pseudo-random generator based on discrete chaotic iterations,” INTERNET’09, 1-st International conference on Evolving Internet, Cannes, France, August 2009, PP 71–76.

[16] J. Bahi, C. Guyeux, and Qianxue Wang, ”A pseudo random numbers generator based on chaotic iterations; Application to watermarking,” International conference on Web Information Systems and Mining, WISM 2010, vol. 6318 of LNCS, Sanya, China, October 2010, PP 202–211.

[17] Y. Hu, X. Liao, K. W. Wong, and Qing Zhou, “A true random number generator based on mouse movement and chaotic cryptography,” Chaos, Solitons & Fractals, vol.40, no. 5, 2009, PP 2286–2293.

[18] S. Behnia, A. Akhavan, A. Akhshani, and A.Samsudin, “A novel dynamic model of pseudo random number generator,” Journal of computational and Applied Mathematics –Journal of Computer and Appl. Math, vol. 235, no. 12, 2011, PP 3455-3463.

[19] Adi A. Maaita, Hamza A. A. Al_Sewadi, Abdulameer K. Husain, and Osama M. Al-haj, “A cryptographically secure Multi-stage pseudo-random number generator”, International Journal of Applied Research in Computer and Communication Engineering IJARCCE, Vol. 4, Issue 5, May 2015, DOI 10.17148/IJARCCE.2015.4503, PP 12-18.

[20] Adi A. Maaita, Hamza A. A. Al_Sewadi, “Deterministic Random Number Generator Algorithm for Cryptosystem Keys”, World Academy of Science, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:9, No:4, 2015, PP 972-977.