International Journal of Information, Control and Computer Sciences
World Academy of Science, Engineering and Technology
ISSN: 2517-9942
Vol:4 2010-02-23
Vol:4, No:2, 2010

A Case Study of Key-Dependent Permutations in Feistel Ciphers*

Hani Almimi[1], Ola Osabi[2], and Azman Samsudin[3]

***Abstract*—**Many attempts have been made to strengthen Feistel based block ciphers. Among the successful proposals is the key-dependent S-box which was implemented in some of the high-profile ciphers. In this paper a key-dependent permutation box is proposed and implemented on DES as a case study. The new modified DES, MDES, was tested against Diehard Tests, avalanche test, and performance test. The results showed that in general MDES is more resistible to attacks than DES with negligible overhead. Therefore, it is believed that the proposed key-dependent permutation should be considered as a valuable primitive that can help strengthen the security of Substitution-Permutation Network which is a core design in many Feistel based block ciphers.

Keywords **—**Block Cipher, Feistel Structure, DES, Diehard Tests, Avalanche Effect.

I. Introduction

MANY countermeasures are needed to protect data. Cryptography is one of the most important techniques used to protect data while the data is being transmitted or even stored in storage devices. Cryptography is the science of creating encryption and decryption algorithms, while Cryptanalysis is the art of attacking the encryption system. The aim of such attack is to recover the secret key using two methods: brute-force attack and cryptanalytic attack. Nowadays both methods are used to verify the strength of any newly proposed encryption algorithm [1]–[3].

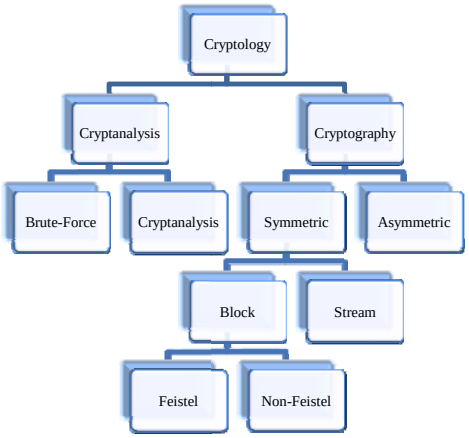

Fig. 1 Simple classification of Cryptography

[1] Hani_Mimi@yahoo.com
[2] Ola_osabi@yahoo.com
[3] Azman@cs.usm.my
* This work has been funded by Universiti Sains Malaysia through research grant 1001/PKOMP/817002

Cryptographic systems can be classified into transposition and substitution, symmetric and asymmetric, and block and stream cipher [1], [3]. In symmetric algorithms, one secret key is used by both parties for encryption and decryption. Whereas two keys, private and public, are used for encryption and decryption in asymmetric algorithms. In general asymmetric cryptography depends heavily on number theory, whereas modern symmetric algorithms which are considered as product cipher, depends on substitution and transposition [1]. Stream cipher encrypts/decrypts a stream of bits or bytes; bit by bit or byte by byte at a time. The block cipher encrypts/decrypts a plaintext block of *N* bits and produces a ciphertext block of the same size. Block ciphers can be classified into Feistel-base such as DES or non-Feistel such as AES [1], [2]. A simple and incomplete taxonomy of Cryptography techniques is depicted in Fig. 1.

Fig. 2 depicts one round of Feistel structure. In general, Feistel structure is composed of *R* rounds, for example 16 rounds in DES. All rounds have the same structure. In each round a subkey is generated to be used inside the function *F*. The output from function *F* is combined with the left half from the previous round using XOR operation. At the end of the round, the two halfs are swapped, the left half will be the right half, and vice versa [1]. For more details on Feistel structure refer to [1]–[3].

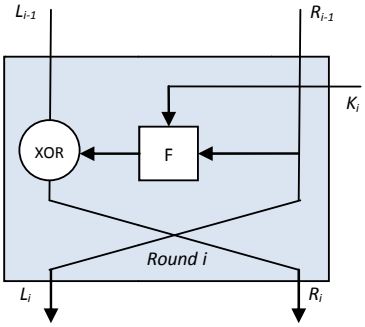

Fig. 2 One round in a Feistel structure

The combination of transposition and substitution operations is called substitution-permutation network (SPN) [1]. Feistel network has many features and parameters that one should consider when inventing or modifying Feistel-base cipher. Some of these parameters and features are: number of rounds, subkey generation algorithm, and round function. In general more rounds mean higher security. Also, if the round function is more complex then to cryptanalysis the cipher will be harder too. It is also mentioned in [1] that there are other two considerations in Feistel cipher design: the first is the

speed of encryption and decryption and the second consideration is the ease of analysis.

The goal of the function *F* is to produce a confused and diffused block. Confusion is achieved through the using of substitution boxes (S-box) and the diffusion is achieved by the permutation boxes (P-box). The aim of confusion and diffusion in block ciphers is to resist the statistical cryptanalysis. Diffusion can be achieved by applying permutation repeatedly on data followed by a function that can remove any statistical relation between the plaintext and the ciphertext. The operation of removing the relationship between the key and the ciphertext is called confusion. Confusion can be achieved by applying non-linear substitution to the encryption operation [1].

Data Encryption Standard (DES) is the most famous symmetric encryption algorithm that is based on Feistel structure. It was a standard for about 20 years. DES was considered not secure in 90's. Consequently, new algorithms were needed to replace DES. One of the solutions was Multiple DES, which was later called Triple DES (3DES) [*1*]. 3DES can be used with two keys or three keys. 3DES is still being used, since the cost to brute-force attack 3DES is still relatively high, 2^{112}.

In this paper, DES was chosen as the case study since it satisfies the following criteria: technically, it is still being used in 3DES. It is based on Feistel structure. It is one of the strongest algorithms against known attacks. It is not complicated and can be analyzed easily. And it is also relatively fast in encryption and decryption [1]–[3].

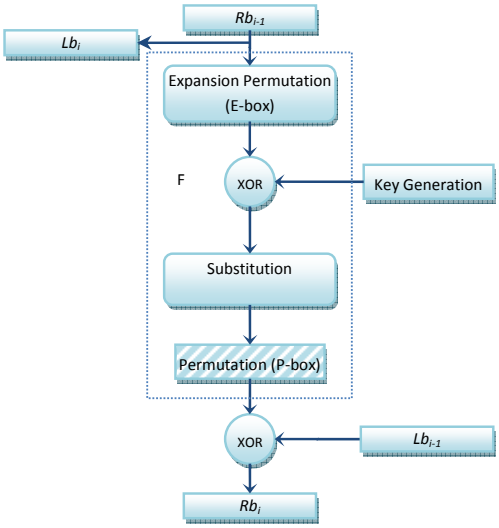

Fig. 3 One round of DES

In each round of DES (see Fig. 3), the sub-keys of 48 bits each are generated from the original key by using operations such as shifting, permutation and selection. An expansion permutation (E-box) is applied to the right half of the block. Then the result is combined with the subkey using XOR. After that, 8 S-boxes are used to produce a 32 bits output which will be permutated using the permutation box P-box. The above operations, which are known as round, are repeated 16 times. There are other two permutations, initial and final permutations, which applied to the input block before the first round and to the output block after the last round. These permutations are applied outside the function *F*. These permutations are not considered in this paper since they do not affect the differential cryptanalytic attack on DES [3]. The first permutation inside *F* is the Expansion Permutation Box (E-box). The goal of E-box is to achieve the avalanche effect as quickly as possible, whereas, the goal of the S-boxes is to strengthen the security of DES by using non-linear operations. Finally, the P-box is a static permutation which adds extra randomness to the output of *F* in order to achieve more robustness against known attacks. TABLE I shows the contents of P-box.

TABLE I
DES PERMUTATION BOX (P)

Permutation Function (P)							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Any attempt to reduce the number of rounds in DES will weaken the algorithm. It has been showed that any version of DES with fewer rounds can be broken with a known-plaintext attack easier than by brute-force attack [*3*]. On the other hand, increasing the number of rounds will increase DES resistance to differential cryptanalysis at the expense of more processing time.

There are many attempts to enhance DES. In one method [3], instead of generating the subkeys in the conventional way, they used different independent keys. Although the linear cryptanalysis of this method was greater than DES, it can be broken using differential cryptanalysis with 2^{61} chosen plaintexts.

Another variant of DES, called DESX [3], uses two keys; one is the normal DES key, which is 56 bits long, while the other is called whitening key with 64 bits long. The whitening key is combined with the plaintext before the first round. A key computed from both keys, 56-bit key and 64-bit key, is XORed as a one-way function to produce the final ciphertext. The complexity in terms of differential cryptanalysis is 2^{60} known plaintexts and 2^{61} of chosen plaintexts.

Generalized DES [3] is a new attempt to enhance DES. The number of rounds in addition to the block size was varied. However, it is not more secure than original DES [3].

Many attempts had also been done to enhance S-boxes. Some changed the order of S-boxes, while others tried to change their contents. On the other hand, it has been shown that DES S-boxes were optimized to resist differential and linear cryptanalysis [3]. Because of the DES S-box size, random or permuted S-box is easier to break. However, in s^n S-boxes [3] a new structure of the first two S-boxes of the eight S-boxes in DES were presented. They were more secure against differential and linear cryptanalysis. Another attempt was to use key-dependent S-boxes. But in this method, a

procedure must be followed in order to produce secure S-boxes against cryptanalysis. As mentioned earlier, it has been shown that one cannot create a random key-dependent DES S-boxes because this will weaken the algorithm [3].

Block cipher algorithms were classified into two types [4]: fixed structure block ciphers and key-dependent block ciphers. In DES, a fixed P-boxes and S-boxes are used. Khufu and Blowfish are examples of key-dependent algorithms [4]. In Khufu, despite it uses a secret key-dependent S-boxes its avalanche effect is achieved only at the 8^{th} round. Khufu is also vulnerable to the "meet in the middle" attack. In addition, the process of key generation is complicated and slow [3].

Differential and linear cryptanalysis strongly affects the development of block ciphers. So many invented methods tried mainly to resist such cryptanalytic attacks [4]. One of the invented methods that resisted cryptanalysis is DSDP [3]. A variable key length is used in DSDP utilizing the Feistel structure. DSDP concentrated on the performance by trying to build a fast key-schedule algorithm. Key-schedule, according to their definition, means subkey, S-box and P-box generation. DSDP was compared against Twofish and Khufu [4]. The number of rounds was not mentioned clearly. However, it is claimed that DSDP performance was reasonable because the number of rounds was smaller than the standard [4].

From the previous discussed methodologies, it can be seen that key-dependent S-boxes and key-dependent P-boxes is one of the promising methods in enhancing block ciphers. However, none of the researchers had addressed modification on the permutation box inside the Feistel structure. What will be the effect of changing the permutation box contents on the Feistel structure? Will it enhance the security or degrade it. Will the avalanche effect be affected by the modification? In order to answer these questions, we had proposed a new key-dependent P-box to be tested on DES. In our case study, the DES S-boxes were not modified since they were optimized to resist the cryptanalysis.

II. Proposed Work

A key-dependent permutation box (P_k-box) is proposed. Whenever the key is changed a new P_k-box will be generated according to the key. The P_k-box will not be calculated in each round as it was addressed by many papers [3], [4]. We focus on fast and strong permutation similar to what was found in RC4 algorithm [5]. Rather than creating a new algorithm such as in [4], or other methods that are summarized in [3], we decided to improve one of the technically strongest encryption algorithms ever known, DES. We tested the effectiveness of the P_k-box on DES.

The RC4 is a fast stream cipher algorithm that is used in standards such as Secure Sockets Layer/Transport Layer Security (SSL/TLS) and IEEE 802.11 wireless LAN. It is based on using random permutations. The period of RC4 is greater than 10^{100} and it is highly nonlinear. The RC4 algorithm is considered secure algorithm [1]. It is also mentioned in [3] that it is resistible against differential and linear cryptanalysis. It uses variable key size to initialize the state vector *S*. The key is copied to another vector *T*. Then an initial permutation is applied to *S* using *T*. Then the entries of *S* are swapped, such that *S[i]* is swapped with *S[j]*, where *i* is the iteration variable, from 0 to 256, and *j* is computed using the equation "*(j+S[i] + T[j]) mod 256*". The aim of the pervious steps is to make the initial permutation. The next phase is generating a byte *K* using some mixing, swapping and permutation that will be XORed with second byte of a plaintext [1], [3].

Operations found in RC4 were customized to be used in this paper. The size of the vectors was modified to be 32. The same process of initializing the vector *S* was also used. The DES key was used to initialize the vector *T*. Then a loop was used to go through *S* and make the swapping which results in the initial permutation. The values of *S* were used to setup the P_k-box of DES. For simplicity, the new modified DES will be called MDES. The following is the algorithm that was used to initialize the P-box:

Algorithm of P_k-box initialization

```
Initialize S with the values 0,1,…, 31
Put DES key inside T
Do the permutation of S as stated in RC4
Then S is used to setup the Pk-box
```

III. Implementation and Results

DES was chosen as a case study since it has many good characteristics as mentioned earlier. DES and MDES were implemented in C++ and compiled using MS Visual C++ v6.0. Three tests had been done in order to measure the effect of key-dependent permutation on Feistel structure algorithms such as DES. The first test is Diehard Tests which measures the randomness within a data sequence. The second test was to measure the avalanche effect in MDES and compare it to DES. The last test was the performance test. The test was performed in order to measure the performance of MDES compared to DES. More details on these tests are mentioned in the following sections.

A. Diehard Tests

Diehard Tests are a set of statistical tests, in a software package, that were designed to identify weaknesses in many common non-cryptographic pseudorandom number generator (PRNG) algorithms. Nowadays, Diehard Tests become well-known in the area of cryptography. It is used to analyze the output of a random number generator in order to evaluate their randomness. The result of Diehard Tests shows, in terms of randomness, if the algorithm is accepted or not. Statistical tests may be used to evaluate the strength of a cipher. It is specially used in stream ciphers since they use a PRNG to generate a sequence which then combined by using XOR operation, with the plaintext to produce the ciphertext. The sequence must be random as much as possible in order to get a better resistance to cryptanalytic attacks [6]–[9]. Furthermore, randomness tests were used to measure the strength of block ciphers [6]. There were many block cipher algorithms which were examined by statistical tests [10], [11].

Diehard suite operates on a single large file (11 Mega bytes) produced by the algorithm. The Diehard Tests consist of 18

tests to assess the randomness or a sequence as mentioned earlier. The Diehard Tests produce values between zero and one (0-1). Each result is called "p-value". These p-values should be greater than 0.01 and smaller than 0.99 in order to consider the results as accepted which leads to accept the randomness of the provided sequence with a confidence level of 99% [5, 12, 13]. Unacceptable PRNGs will produce p-values that are within 0.00001 of zero or one. The p-values can tell that the sequence passed the related test or not, but there is no way to compare these p-values which lie in the success interval [14].

There are other statistical tests for random number generators such as NIST Test Suite [15], FIPS 140-X [2], [16], Knuth's Test Suite, the test of Vattulainen, Ala-Nissila, Kankala [17] and TestU01 [18]. These statistical tests are needed to make sure that the ciphertext produced by an algorithm cannot be statistically attacked. Those tests are important in measuring the acceptance level of random number generators [19]. They also can be used to judge the output of stream cipher algorithms [13]. If the output of a stream cipher is purely random and it passes the statistical test, then the algorithm is considered secure. This case can be applied to block cipher too. But in order to do that, a large amount of plaintext must be encrypted by many keys which were generated randomly. Varying the number of rounds in DES affected the randomness of the produced sequence of ciphertext as mentioned in [20]. The DES with full rounds will be considered here. The plaintext was divided into segments, and each segment was encrypted using different key to output a single ciphertext file. We had two cases, either to randomize the production of the plaintext or to randomize the production of the keys. The first case is not true since data is not random in nature even though this situation can happen if the input plaintext to an encryption algorithm is compressed data or previously encrypted data. In the second situation, one can encrypt a message by a key and the next one by another key and so on which is hard to happen. Still, encrypting different messages by different keys will be easier if it is automated by a random number generator. As a result, the second case was chosen to be implemented and tested.

Diehard Tests were used in order to measure the randomness of both DES and MDES output. As a result of those tests, the resistance to statistical attacks was measured. Both DES and MDES were subjected to the same tests and also fed with the same plaintext. Five experiments had been conducted on both algorithms for each algorithm by varying the number of blocks and the number of keys. A file of size (11 Mega bytes) was needed for the Diehard Tests. So, the plaintext file was encrypted in the first experiment by both algorithms with 10,000 blocks per each different key. The block size was 8 bytes. This means that the first 10,000 blocks of the plaintext file was encrypted with the key k_1, then the following 10,000 blocks was encrypted using a different key (generated randomly) k_2, and so on. The second experiment was done using 1,000 blocks per each different key, the third experiment with 100 blocks per key, the fourth experiment with 10 blocks per key, and finally the last experiment was done with 1 block per key.

TABLE II
THE RESULTS OF DIEHARD TESTS ON DES AND MDES

Test Name	Blocks per DES-Key									
	10000		1000		100		10		1	
	MDES	DES	MDES	DES	MDES	DES	MDES	DES	MDES	DES
BIRTHDAY SPACINGS	F	F	F	F	F	F	F	F	F	P
OVERLAPPING 5-PERMUTATION	P	F	F	F	F	F	P	F	P	P
BINARY RANK-31	F	F	F	F	F	F	F	F	P	P
BINARY RANK -32	F	F	F	F	F	F	F	F	P	P
BINARY RANK - 6x8	F	F	F	F	F	F	F	F	P	P
BITSTREAM	F	F	F	F	P	P	P	P	P	P
OPSO	F	F	F	F	F	F	F	F	P	P
OQSO	F	F	F	P	P	P	P	P	P	P
DNA	P	P	P	P	P	P	P	P	P	P
COUNT-THE-1's on a stream of bytes	F	F	P	F	P	P	P	P	P	P
COUNT-THE-1's for specific bytes	F	F	P	P	P	P	P	P	P	P
PARKING LOT	F	F	F	F	F	F	P	P	P	P
MINIMUM DISTANCE	F	F	F	F	F	F	F	F	F	F
3DSPHERES	F	F	F	F	F	F	F	F	P	P
SQEEZE	P	F	P	P	P	P	P	F	P	P
OVERLAPPING SUMS	P	P	P	P	P	P	P	P	P	P
RUNS	P	P	P	P	P	P	P	P	P	P
CRAPS	P	P	P	P	P	P	P	P	P	P
Total success	6	4	7	7	9	9	11	9	16	17

The results of the previous experiments are summarized in TABLE II. As it can be seen from the table, MDES and DES failed most of the tests for the first experiment (10,000 blocks per different key). MDES passes one test more than that for DES. All passed tests were passed with a confidence level of 99% within the success range [0.01 - 0.99]. In the second and third experiments, both algorithms passed the same number of tests. In the fourth experiment (100 blocks per different key) MDES passed 11 tests while DES passed only 9 tests. Finally, the last experiment (1 block per different key) showed that DES passed 17 tests while MDES passed only 16 tests. As a result from the previous experiments, it can be concluded that MDES passed more tests than DES. This means that MDES exhibits more resistance to statistical attacks which can be proved by the results of the Diehard Tests.

B. Avalanche Effect

The avalanche effect means that a small change in the plaintext or the key should produce a huge change in the ciphertext. DES has been proved to have a strong avalanche effect [1]. So, it is necessary to verify that MDES will give at least the same results as DES. Two tests for MDES were conducted to measure the avalanche effect. The first test was conducted using a fixed key and two plaintexts which were different only in one bit. The following is the key that was used:

11001000 00111111 10101001 00100110 10101110 11011011 10100111 11100100

The first plaintext that was used was

01000001 01000001 01000001 01000001 01000001 01000001 01000001 01000001

And the modified plaintext was

01000001 01000001 01000001 01000001 01000001 01000001 01000001 01000010

The second test was conducted using two keys with the same plaintext: the first key was

11001000 00111111 10101001 00100110 10101110 11011011 10100111 11100100

And the second key was

01001000 00111111 10101001 00100110 10101110 11011011 10100111 11100100

And the plaintext was

01000001 01000001 01000001 01000001 01000001 01000001 01000001 01000001

TABLE III
THE AVALANCHE EFFECT OF DES AND MDES

round	Number of bits that differ			
	Two plaintexts with one key		One plaintext with two keys	
	MDES	DES	MDES	DES
1	6	6	2	2
2	21	15	10	15
3	31	27	13	26
4	35	34	21	22
5	35	38	33	27
6	32	33	34	30
7	36	32	28	24
8	37	34	27	26
9	39	32	34	29
10	34	34	33	36
11	28	35	32	34
12	32	38	34	28
13	31	33	33	33
14	36	35	36	35
15	38	36	41	34
16	31	34	39	27

The results are summarized in TABLE III. It is clear from the results that, when the plaintext was varied with a fixed key, MDES is better than DES because the number of bits that had been changed after the second and third rounds are more than DES. On the other hand, when the plaintext was fixed and the key was changed, modified by one bit, DES was better than MDES in rounds 2, 3 and 4. But after the fourth round MDES showed a better avalanche effect than DES. As a result, both showed a strong avalanche effect. The average of bits changed after the last round in MDES was 35 while the average in DES was 30.5. This shows that the avalanche effect that can be achieved using MDES is almost the same as that can be achieved using DES.

C. Performance Test

TABLE IV
THE ENVIRONMENT SPECIFICATIONS FOR THE PERFORMANCE TEST

PC	Operating system	Processor	Memory	Compiler
1	Windows XP Professional (5.1, Build 2600) Service Pack 2	Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz (2 CPUs)	2046MB	MS Visual C++ 6
2	Windows XP Professional (5.1, Build 2600) Service Pack 3	Intel(R) Pentium(R) 4 CPU 1.80GHz	512MB	MS Visual C++ 6
3	Windows XP Professional (5.1, Build 2600) Service Pack 2	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz	1024MB	MS Visual C++ 6
4	Windows XP Professional (5.1, Build 2600) Service Pack 3	AMD Phenom(tm) 9650 Quad-Core Processor, MMX, 3DNow (4 CPUs), ~2.3GHz	3.5 GB	MS Visual C++ 6

Three tests had been conducted in order to measure the performance of the modified algorithm compared to the original one. TABLE IV shows the specifications of the machines that were used to test DES and MDES. The same 1,000,000 plaintext blocks of 8 bytes each were encrypted by both algorithms using 100 different keys. The keys were generated randomly every 10,000 blocks of plaintext. The previous values related to the program specifications were summarized in TABLE V.

TABLE V
THE PARAMETERS FOR THE PERFORMANCE TEST

Parameter	Value
Block Size	64 bits
The Number of iterations	1000,000
Number of blocks processed	1000,000
Number of keys generated and used	100

The time taken by each algorithm on each machine is shown in TABLE VI. The average in the last row of TABLE VI showed the percent of which MDES was slower than DES. MDES was slower than DES with an average of 1.97%, which can be tolerated since it did not affect the performance so much. The formula used to compute the percentage of difference of algorithm speed is as follows:

Difference in Percent = 1- (DES result / MDES result) × 100%

TABLE VI
RESULTS OF PERFORMANCE TEST

ALGORITHM	TIME TAKEN IN MILLISECOND (MS)			
	PC 1	PC 2	PC 3	PC 4
DES	80953	158812	74046	75640
MDES	81734	161968	75718	77796
DIFFERENCE (MS)	781	3156	1672	2156
DIFFERENCE (%)	0.96%	1.95%	2.21%	2.77%

IV. CONCLUSION

A key-dependent permutation box (P_k-box) for Feistel structure algorithms, such as DES, has been proposed. The P_k-box of MDES was modified according to the secret key. Each time the key changes, the P_k-box will change accordingly. The goal of the modification is to enhance the confusion and

diffusion of the Feistel structure; in addition to enhance the avalanche effect for MDES. By making the P_k-box key-dependent, the internal structure of the algorithm will be more secure since the contents of the P-box are hidden and related to the secret key. The algorithm will also become more resistible to differential and linear cryptanalysis in addition to statistical attacks. A final consideration was to make sure that the performance of the algorithm is not degraded too much by the modification.

Three tests were conducted to ensure that the goals have been accomplished. The Diehard Tests was conducted in order to measure the randomness of the ciphertext of both DES and MDES. The results showed that MDES passed more tests than DES. This means that MDES was more immune to statistical attacks than DES. Two more tests were conducted to measure the avalanche effect of both algorithms. Both DES and MDES indicated a strong avalanche effect. However, 35 bits were changed in average after the last round in MDES while only 30.5 bits in DES. Thus, DES and MDES exhibit a good avalanche effect. Finally, four performance tests were conducted. MDES was slower than DES with an average of 1.97%, which can be tolerated. This average leads to a conclusion that the overhead is negligible**.**

REFERENCES

[1] W. Stallings, *Cryptography and Network Security Principles and Practices*. Prentice Hall, 2006.

[2] A. Menezes, P. Van Oorschot, and S. Vanstone, *The Handbook of Applied Cryptography*. CRC Press, Inc., 1996.

[3] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 1996.

[4] R. Zhang and L. Chen, "A Block Cipher Using Key-Dependent S-box and P-boxes," in *IEEE International Symposium on Industrial Electronics, ISIE 2008*, pp. 1463 - 1468.

[5] The Florida State University, The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness. [Online]. http://www.stat.fsu.edu/pub/diehard/

[6] V. Katos, "A Randomness Test for Block Ciphers," Applied Mathematics and Computation, Vol. 162, p. 29–35, 2005.

[7] M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "A Double Scroll Based True Random Bit Generator," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS'04)*, Vancouver, BC, Canada, 2004, p. 581–584.

[8] I. Jang and H. S. Yoo, "Pseudorandom Number Generator Using Optimal Normal Basis," *in Proc. International Conference on Computational Science and its Applications*, Glasgow, Royaume-uni, 2006, vol. 3984, pp. 206-212.

[9] D. Bucerzan, "A Cryptographic Algorithm Based on a Pseudorandom Number Generator," in *10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2008, pp. 453-456.

[10] P. Hellekalek, and S. Wegekittl, "Empirical Evidence Concerning AES," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, p. 322–333, 2003.

[11] J. Soto, "Statistical Testing of Random Number Generators," in *Proceedings of the 22nd National Information Systems Security Conference*, Virginia - USA, 2000.

[12] X. Zhangy, K. Tangy and L. Shuz, "A Chaotic Cipher Mmohocc and Its Randomness Evaluation," in *International Conference on Complex Systems (ICCS2006)*, Boston, 2006 [not published].

[13] K. M. A. Suwais, "Parallel Platform For New Secure Stream Ciphers Based On NP-HARD Problems", Phd thesis, Uinversity Sains Malaysia, 2009.

[14] S. H. Lee, H. Y. Jeong and Y. S. Lee, "Application-Adaptive Pseudo Random Number Generators and Binding Selector," in *The 23rd International Technical Conference on Circuits/Systems, Computers and Communications*, Shimonoseki - Japan, 2008, pp. 1561 - 1564.

[15] F. Pareschi, R. Rovatti and G. Setti, "Second-level NIST Randomness Tests for Improving Test Reliability," in *IEEE International Symposium on Circuits and Systems*, NewOrleans, 2007, pp. 1437-1440.

[16] "FIPS 140-1 Non-Proprietary Cryptographic Module Security Policy," 2000. http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140sp/140sp111.pdf. visited in October, 2009.

[17] I. Vattulainen, T. Ala-Nissila and K. Kankaala, "Physical Tests for Random Numbers in Simulations," *Phys. Rev. Lett. 73*, 19, 2513–2516. 1994.

[18] K. H. Tsoi, K. H. Leung and P. H. W. Leong, "High Performance Physical Random Number Generator," *IET computers & digital techniques*, pp. 349-352, 2007.

[19] K. H. Yalcin, J. A. K. Suykens and J. Vandewalle, "True Random Bit Generation From a Double-Scroll Attractor," in *IEEE Transactions on Circuits and Systems I*, 2004, pp. 1395 - 1404.

[20] F. A. Feldman, "A New Spectral Test for Nonrandomness and the DES," in *IEEE Transactions on Software Engineering*, 1990, pp. 261 - 267.

[21] A. G. Konheim, *Crytography: A Primer.* Wiley Interscience, 1981.