

Lightweight Cryptosystem for Image Encryption Using Auto-Generated Key

¹Mohammed A. Fadhil Al-Husainy, ¹Hamza A. Al-Sewadi and ²Shadi R. Masadeh

¹Faculty of Information Technology, Middle East University, P.O. Box 383, 11831 Amman, Jordan

²Faculty of Information Technology, Isra University, Amman, Jordan

Abstract: Due to the lengthy and heavy computations required by the standard and powerful cryptosystems such as AES and 3DES and the need for efficient and secure cryptosystems for use over the cloud computing environment, a new era of lightweight cryptographic algorithms for security has emerged, especially for applications with limited resources such as remote sensors, smart cards and embedded healthcare devices. This study proposes a block cipher lightweight image cryptosystem that uses any digital file as a seed for generating the secret key. It works on a 32 bit block size and a key of any length and type of digital data. It generates a 16×16 substitution table of bytes, hence, a key space of 2048 is used, besides part of the key is embedded into the encrypted image. Experimental results have demonstrated good security level and a clear improvement in the encryption time when it is compared with other widely used systems.

Key words: Symmetric block cipher, image processing, key embedding, cloud computing, digital data

INTRODUCTION

The security of transmitted data and data at rest is becoming the real hazard for computer clouds applications. The customer's worries of data base systems utilizing the cloud for their bulk data storage are increasing due to the ever increasing processing speed and development of advanced software capabilities. This increased computer efficiency empowers hackers and intruders to breach security measures, hence, new and more powerful encryption techniques have to be regularly developed. For example, powerful cryptosystems such as the Data Encryption Standard (DES) (Schneier, 1996) which was secure enough, since, its approval by the National Institute of Standard and Technology (NIST) (Stallings, 2011) in 1976. It encrypts the data as blocks of 64 bits length using a key of 56 bits length. DES was secure enough till around the end of the twentieth century as it has become vulnerable for attacks due to the advancement in the growing computation efficiency of modern computers, then it was replaced by 3DES (Triple DES) (Alanazi *et al.*, 2010; Aleisa, 2015) which can be operated with two or three keys. Triple DES is highly secure till now, however, it requires long encryption and decryption times which might not be acceptable for most applications. Hence, the Advanced Encryption Standard (AES), also, known by its original name Rijndael was developed and adopted as the replacement for DES. It has three different versions with key lengths of 128, 192 and 256 bits. The 3DES and AES cryptosystems have proved their strength and have been in use for some time for high

secrecy and sensitive applications for encrypting various multimedia such as text, audio and images, however, they are criticized of being not suitable for applications with limited resources such as remote sensing stations, RFID, smart cards, etc. Moreover, for still images and video cryptography (Al-Husainy and Al-Sewadi, 2017; Al-Husainy, 2006; Bhowmik and Acharyya, 2011) due to the typically, huge size of these files, extremely long time for encryption and decryption is required and may not be tolerated by most applications. Hence, for such application as well as for less sophisticated applications, that needs secure but heavy data transit or bulky storage and retrieval of data such as databases utilizing the storage capacity of computer cloud, they still require some classical encryption techniques that are fast enough with acceptable security level such system may be referred to as lightweight cryptosystem. Designers of such cryptosystems look for a balance between three crucial factors: security strength, financial cost and efficiency. This study presents an encryption/decryption algorithm which could be considered as lightweight and serve this purpose.

Literature review: So, many lightweight symmetric cryptosystems have been developed and reported for appropriate applications. A good survey and comparison for such cryptosystems is reported by Kushwaha *et al.* (2014). This study lists some of the most important and widely used such as HIGHT, LBlock, DESL, DESXL, CLEFIA, PRESENT, LED, TWINE, RECTANGLE, KLEIN, SIT.

Hong *et al.* (2006) proposed a 64 bits block size lightweight cryptosystem with a 128 bits key length, iterated in 32 rounds modified Feistel network having two type of operations, XOR operation combined with left or right rotations. Its design was suited for hardware implementation on ubiquitous devices such as wireless sensor nodes and RFID tags having almost the same chip size as AES but works much faster. Its security was tested by a differential attack and was slightly lower than the exhaustive search.

In 2007, CLEFIA-128 symmetric block cipher developed by Sony as reported by Shirai *et al.* (2007) is designed to be suitable for both hardware and software. It encrypts data block of 128 bits under 128 bits key length and 28 rounds with Feistel structure. There are other CLEFIA Versions under 192 or 256 bits key lengths with 22 and 26 rounds, respectively. It implements 2 different 8 bits S-boxes followed by a diffusion matrix multiplication inspired from the AES mix columns operation. Using the impossible differential attack against CLEFIA reduced to 12 rounds for a 128 bits key in a time complexity of 2119 encryptions, refer to Tsunoo *et al.* (2008). Obviously, the time complexity increases for longer keys.

Two lightweight variants of the data encryption, namely DESL and DESXL were proposed by Leander *et al.* (2007). For DESL, single S-box is used instead of different ones and no initial and final permutations whereas for DESXL, a whitening step is implemented to enhance the security using 184 bits key length. According to the researcher claim, no attack has been exhibited against DESL and DESXL.

Bogdanov *et al.* (2007) presented an ultra-light block cipher named "Present". It was an example of Substitution-Permutation Network (SPN) structure having a block size of 64, 80 or 128 bits key size with 31 rounds with multiple uses of 4 bits S-box. PRESENT is optimized for hardware implementation.

Cheng *et al.* (2008) presented a novel compact evolutionary SPN structure lightweight that implements the same data path for both encryption and decryption. It is designed for hardware implementation to be embedded on digital systems such as programmable gate arrays, however, it was less complex than "Present" but with high throughput. It also has block size of 64 bits but with 128 bits key.

De Canniere *et al.* (2009) reported an efficient hardware-oriented block ciphers. It processes block size of 32, 48 or 64 bits 80 with a key of 80 bits length. Two iterative algorithms were suggested, KATAN and KTANTAN differing only in the key scheduling and

using two Boolean functions with output shifting with 254 iterations in total. Both algorithms, KATAN and KTANTAN have a serialized structure.

Knudsen *et al.* (2010) presented a lightweight block cipher for integrated circuit printing. It also has SPN structure with 48 or 96 bits block size with 80 or 160 bits key lengths. The cryptosystem is suitable for a low cost personal encryption/decryption circuits.

Wu and Zhang (2011) presented a Feistel structure lightweight cipher, "LBLOCK" which also has SPN structure and efficiently implemented in both software and hardware. It is designed for a block size of 64 bits with 80 bits key length and 32 rounds runs. LBLOCK used variant-Feistel structure.

The lightweight block cipher, LED is suggested by Guo *et al.* (2011). It provides a reasonable performance efficiency for software implementation. It encrypts 64 bits blocks and uses four key sizes, namely 64, 80, 96 and 128 bits. The S-box of PRESENT cipher is implemented in the execution of LED cipher.

Gong *et al.* (2011) reported an SPN structure lightweight cryptosystem. They claim it's advantageous in software performance on legacy sensor platforms, besides compact implementation for hardware application. It used 16 S-boxes (of 4 bits elements) for substitution and Rotate and Mix-Nibbles for permutation. Klein ciphers 64 bits block using 64, 80 or 96 bits key length with 12, 16 or 20 rounds, respectively.

Suzaki *et al.* (2013), developed a generalized Feistel structure, multiple platform cipher system named "Twine". They claim that it is extremely-small hardware and quite efficient on embedded software. Twine has block size of 64 bits for 36 rounds with key length of either 80 or 128 bits length. Each round of Twine involves a nonlinear substitution layer using 4 bits S-boxes and 4 bits block permutation layer.

Zhang *et al.* (2014) proposed a bit-slice ultra-lightweight block cipher named "Rectangle". It is found suitable for multiple platforms achieving highly competitive software performance and requires very low area in hardware. Its structure is an SPN designed for 64 bits block size and key length of either 80 or 128 bits, similar to PRESENT cryptosystem but with only 25 rounds.

Masadeh and Al-Sewadi (2017) reported a lightweight segmented block cipher algorithm based on ASCII code maneuver. It has a block size of 60 bits and claimed to be satisfactory for encryption of text messages but not been implemented for image encryption.

In this study, a lightweight cryptosystem is proposed that accepts any digital file as a seed for generating the secret key. It has an SPN structure,

designed for image encryption and uses a 32 bits block size and a key of any length or type: text, image, audio or video. The key file contents are utilized to generate a table of bytes that has a large key space, referred to as substitution table as described next.

MATERIALS AND METHODS

The proposed embedded cryptography algorithm: The algorithm presented here is a new Lightweight block Cipher symmetric cryptographic system (LWC). In brief, LWC consists of the following processes:

- Construct a substitution table from the bytes of the key to be used for performing a substitution operation during the encryption process
- Execute a sequence of steps that include segmenting the input image into segments of four bytes each, coding the image segment using the constructed table
- Extract bits from the bytes of the image segment
- Construct four keys from the extracted bits
- Coding the four keys using the constructed table
- Execute XOR operation between the bytes of the image segment and the four keys
- Finally, arrange the bytes of the resulted image in certain order

The above processes are implemented in both encryption and decryption, however, they obviously differ in their order of execution.

The LCW algorithm: In this study, the suggested Lightweight Cryptographic (LWC) algorithm is included. The important definitions and terminologies are listed first in order to allow for fluent follow up for the steps of the proposed cryptographic system. The generation process of the substitution table, used in the algorithm is described next and then the algorithm and the flowchart of the encryption process are detailed next.

Definitions:

- SImage: the source image or the input image to be encrypted. It is a bitmap color image
- EImage: the encrypted image produced by the cryptographic system (i.e., the encrypted output image). EImage is also a bitmap color image

SKey: The selected secret key by the communicating parties for the generation of the private substitution table which will be used for encryption and decryption. SKey is a digital file of any types such as text, image, audio, video, etc., SKey is treated as a file of bytes.

	0	1	2	...	8	...	13	14	15
0	34	52	78		121			49	125
1	82	47	72		100			42	68
2	64	102	40		45			106	
...									48
10								35	
11	91	105					97		
...									
14	73	59						99	36
15	101	87			83			57	43

Fig. 1: 16×16 table M used for substitution

Generate the substitution table M: The process of generating the two dimensional, 16×16 table M starts by scanning the selected key file SKey in order to fill the table M. The decimal value of the SKey file bytes are taken sequentially to fill the table without repetition. This operation results in the table contents with values between 0 and 255 randomly distributed according to the SKey file content. This table will be used in the performance of substitution operations on the image data and the extracted key during the encryption process. Obviously, each SKey file produces a different distribution of table contents. An example of the substitution table M_i generated by this procedure is illustrated in Fig. 1. It is only partially filled in order to demonstrate the idea of how table M is generated. Obviously, using the same SKey file by sender and receiver will generate the same substitution table M in both sides.

The encryption process: The proposed LWC algorithm is a block cipher that implements two main operations on the bytes of the source image SImage, namely substitution and transposition. Figure 2 depicts a brief flowchart that summarizes the steps followed for the image encryption, followed by the detailed description of all encryption steps.

Substitution: This operation involves data substitution twice using the substitution table M previously generated in addition to an XOR operation as follows.

The source image SImage to be encrypted is first segmented into N segments (S_1, S_2, \dots, S_N) of 32 bits size each, hence, each segment contains four bytes (B_1, B_2, B_3 and B_4):

B_1	B_2	B_3	B_4
-------	-------	-------	-------

where:

$$N = \text{Length of (SImage)}/4 \quad (1)$$

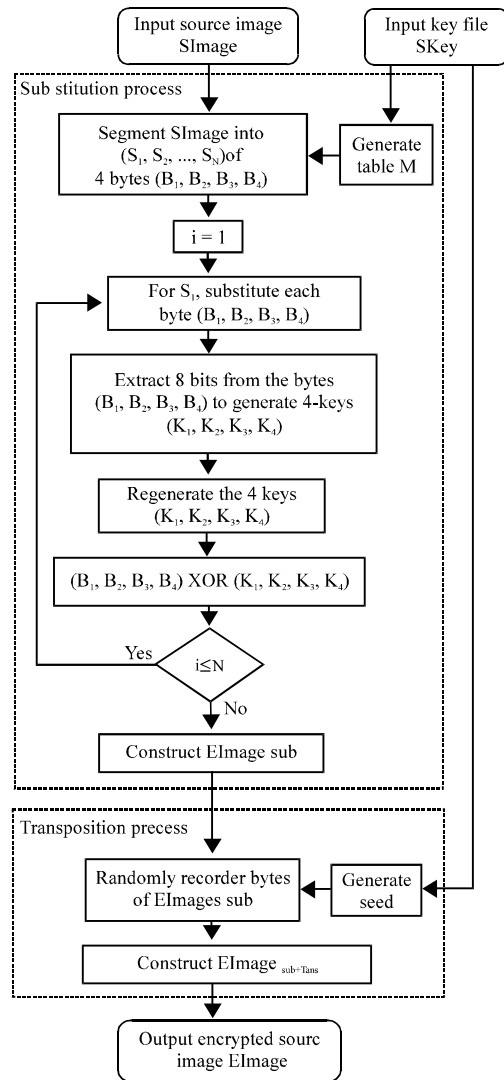


Fig. 2: Flowchart summarizing the encryption process of LWC algorithm

Step 1: For each segment S_i , replace each byte with a new value using the substitution table M to produce a new segment BS_i . Each new byte represents the binary equivalence of the intersection of the row number and the column number of the byte B_i in segment S_i . Where, the row number and the column number, represent the high half and the low half of the new byte, respectively. For example, consider the S_i segment whose bytes are B_1 - B_4 , so, their decimal equivalent numbers are 78, 105, 99 and 101, respectively as follows:

	B1	B2	B3	B4
S_i	78	105	99	101

After using the substitution table M , the resulted BS_i becomes:

K_1 : 00101111	K_2 : 11111000	K_3 : 11001110	K_4 : 10110011
K'_1 : 215	K'_2 : 158	K'_3 : 1014	K'_4 : 113
K''_1 : 48	K''_2 : 83	K''_3 : 35	K''_4 : 97

Fig. 3: The row and column numbers in substitution table M

S_i bytes	B_1	B_2	B_3	B_4
BS_i	00000010	10110001	11101110	11110000
Bites numbers	76543210	76543210	76543210	76543210

Step 2; Use the two bits: 7 and 6 bit of each of the four bytes (B_1 - B_4) of the segment BS_i to form new four bytes that represent four keys (K_1 - K_4). The four keys are generated by cascading these bits in four different permutations as follows:

- K_1 : $B_1(7) B_1(6) B_2(7) B_2(6) B_3(7) B_3(6) B_4(7) B_4(6)$
→ 00101111
- K_2 : $B_4(7) B_4(6) B_3(7) B_3(6) B_2(7) B_2(6) B_1(7) B_1(6)$
→ 11111000
- K_3 : $B_3(7) B_3(6) B_1(7) B_1(6) B_4(7) B_4(6) B_2(7) B_2(6)$
→ 11001110
- K_4 : $B_2(7) B_2(6) B_4(7) B_4(6) B_1(7) B_1(6) B_3(7) B_3(6)$
→ 10110011

Step 3: Find the corresponding byte in substitution table M for each of the four keys (K_1 - K_4). Where, the high and low halves of each key (byte) represent the row and column numbers in substitution table M , respectively as illustrated in Fig. 3.

Step 4: Find the binary representation of the resulted keys in Step 3.

- K'_1 : 48-00110000
- K'_2 : 83-01010011
- K'_3 : 35-00100011
- K'_4 : 97-01100001

Step 5: Perform XOR logical operation for the 4 bytes of block BS_i ($i = 1, \dots, 4$) with the 4 generated keys (K'_1 - K'_4) of Step 4 excluding 6 and 7 bit of each byte of segment BS_i . The operation results into NBS_i as in Fig. 4.

Now, NBS_i represents the encrypted image segment after the substitution operation.

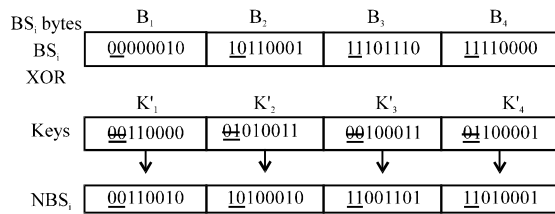


Fig. 4: The operation results into NBS_i

Step 6: Merge all the NBS_i segments to construct a new Encrypted Image EImage_{Sub} which represents the SImage after substitution operation is completed.

Transposition: This operation also, involves two processes an XOR operation and a byte transposition process as follows:

Step 1: Produce a one byte hash value by applying an XOR logical operation among all the bytes of the SKey to be used as a seed for a pseudo-random number generation algorithm.

Step 2: Reorder the Encrypted Image (EImage_{Sub}) bytes randomly according to the pseudo-random number generation algorithm results.

Step 3: Construct a new Encrypted Image EImage_{Sub+Tra} that represents the source image SImage after complete the substitution and transposition operations. This step produces the final Encrypted Image EImage.

The decryption process: To recover the (SImage) original image from the Encrypted Image (EImage) at the receiver side, a decryption process is executed which is similar to the encryption process but in reverse order. The received Encrypted Image (EImage) is first rearranged back to become (EImage_{Sub}) using the randomly generated values by the seed obtained from the Selected Key (SKey). Then segmented into bytes producing 32 bits segments (NBS_i), from which (BS_i) is obtained by XOR'ing with the six least significant bits of keys (K'₁-K'₄) which were already created using the SKey as explained in steps 3-4 of the encryption process. Then using the substitution table M and keys (K₁-K₄) of step 2, the original image segments are obtained from them, the original (SImage) is reconstructed.

RESULTS AND DISCUSSION

Implementation and results: To investigate the LWC algorithm described in this study, hundreds of images were encrypted and decrypted using various types of files as key for each case. Four images of different sizes,

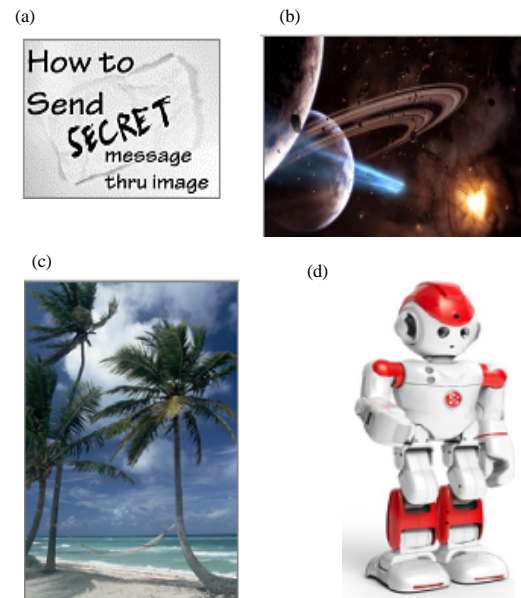


Fig 5: Examples of the images used in the experiments: a) Message (128×85); b) Space (215×320); c) Beach (170×256) and d) Robot (128×234)

Table 1: Key type effects on the encrypted images (robot image)

Key types	PSNR (dB)	Correlation	Encryption time (msec)
Text	9.434	0.205	133
Audio	6.722	0.152	423
Image	4.388	0.102	120
Video	8.174	0.165	736

Table 2-4: PSNR comparison using LWC, DES and AES algorithms

Images	PSNR		
	LWC	DES	AES
Message	6.353	5.975	6.006
Space	4.218	1.922	1.918
Beach	7.357	7.27	7.256
Robot	4.388	5.82	5.786

scenes and color contents, namely: a text, space, beach and robot images (Fig. 5) were included here as examples in order to demonstrate the obtained results.

The Peak Signal to Noise Ratio (PSNR), the cross-correlation and the execution time for encrypting images using different types of files as a secret key, namely: text, image, audio and video multimedia files are computed. Table 1 lists those results for encrypting the Robot image as an example. The differences shown in these measurements are attributed to the differences in the file sizes rather than to the file type itself.

To evaluate the performance of the proposed LWC algorithm, various images were encrypted with LWC, DES and AES encryption algorithms under the same computation environment and the results are compared. The encrypted images for the four selected files are displayed in Table 2. It visually shows the encryption

effect similarities on the images. The histograms for the original images and the encrypted images using the three algorithms are listed in Table 3. They clearly reflect the influence of the encryption processes by noticing the pixels intensity distribution. These histograms indicate that pixels distributions for the Encrypted Images by LWC algorithm gets better and comparable with DES and AES as the image becomes more natural as shown for the beach image. It is worth mentioning that the same key was used for the three algorithms under consideration.

The PSNR, cross-correlation and the processing time for Encrypting the Images using the proposed LWC are calculated for the considered images and listed in Table 4-7, together with those for DES and AES algorithms calculated under the same computation environment for comparison purposes. From Table 2-4 one can notice the following:

In Table 2, the PSNR values for LWC were slightly higher than those for DES and AES algorithms in the cases of the message and the beach images. But in case space image, PSNR values decreased considerably for DES and AES as compared with LWC algorithms. On the other hand for the robot image it increased slightly. These discrepancies might be attributed to the fact that space and robot images were hand drawn with sharp colors and edges and not a natural images.

The cross correlation shown in Table 3 shows minor differences amongst LWC, DES and AES algorithms. These results suggest that the proposed LWC algorithm

Table 3: Cross-correlation comparison

Images	Cross-correlation		
	LWC	DES	AES
Message	0.086	0.084	0.111
Space	0.071	0.075	0.072
Beach	0.076	0.078	0.076
Robot	0.102	0.126	0.116

Table 4: Encryption execution time comparison

Images	Encryption time (msec)		
	LWC	DES	AES
Message	52	175	156
Space	617	1760	920
Beach	147	286	263
Robot	123	169	157

Table 5: Features comparison of various cryptosystems

Encryption method	Block size (bit)	No. of rounds	Key length (bit)	Key space (key)
DES	64	16	56	2^{56}
DESXL	64	16	184	2^{168}
AES-128	128	10	128	2^{128}
AES-192	128	12	192	2^{192}
AES-256	128	14	256	2^{256}
Blowfish	64	16	448	2^{448}
LWC	32	1	256	2^{2048}
PRESENT-80	64	32	80	2^{80}
SIT	64	5	64	2^{64}
HIGHT	64	32	128	2^{128}
PRINT-48	48	48	80	2^{80}
LED-64	64	32	64	2^{64}
KATAN-64	64	254	80	2^{80}
TWINE-80	64	16	80	2^{80}
KLEIN-64	64	12	64	2^{64}
KTANTAN-64	64	32	64	2^{64}
LBLOCK	64	25	80	2^{80}
RECTANGLE-80	64	25	80	2^{80}

Table 6: Encrypted images produced by LWC, DES and AES algorithms






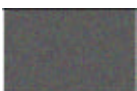











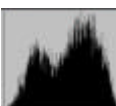

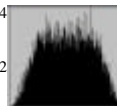



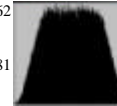



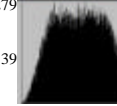
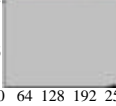
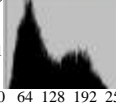
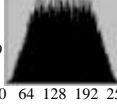
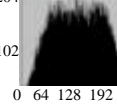
Images	Encrypted image		
	LWC	DES	AES
			
			
			
			

Table 7: Histograms comparison for encryption with LWC, DES and AES algorithms

Images	Original image	Encrypted image		
		LWC	DES	AES
Message	<div>642</div> <div>321</div> 	<div>90</div> <div>45</div> 	<div>81</div> <div>40</div> 	<div>84</div> <div>42</div> 
Space man	<div>5881</div> <div>2940</div> 	<div>1561</div> <div>780</div> 	<div>964</div> <div>482</div> 	<div>962</div> <div>481</div> 
Beach	<div>420</div> <div>210</div> 	<div>286</div> <div>143</div> 	<div>275</div> <div>137</div> 	<div>279</div> <div>139</div> 
Robot	<div>11332</div> <div>5666</div> 	<div>303</div> <div>151</div> 	<div>199</div> <div>99</div> 	<div>204</div> <div>102</div> 
		0 64 128 192 255	0 64 128 192 255	0 64 128 192 255

behaves in a similar way like the other widely used algorithms in the way of protecting the encrypted images.

In Table 4, LWC algorithm has the shortest encryption time for all the four types of images under consideration. LWC algorithm encryption process ranges from about 1.3-3 times faster than that for DES or AES algorithms.

The LWC algorithm may also be compared with DES and AES algorithms in addition to other reported or available lightweight encryption techniques. Such comparison will include block size, number of rounds, secret key length and key space. This comparison is listed in Table 5. The key space in the proposed LWC algorithm is very big due to the use of a very large table for the secret key which is generated using the decimal representation of the key seed contents. Besides, although, one round is used in the suggested algorithm, the number of rounds can be increased if required, however, this is compensated by the large key space complexity.

CONCLUSION

The encryption/decryption approach presented in the algorithm of this study is a lightweight block cipher symmetric cryptosystem that relies on a continuously variable key depending on the message contents and based on secret encryption tables.

The security strength of the proposed algorithms is moderate as compared with the traditional DES and AES cryptographic algorithms, however, the measured processing speed is much faster. Such advantage would be greatly welcomed by many daily applications that involve huge amount of data in database applications utilizing the available and cheap storage on the computer cloud. Therefore, it is anticipated by the researchers that such system would be widely welcomed by such applications.

ACKNOWLEDGEMENTS

Mohammed A. Fadhil Al-Husainy and Hamza A. Al-Sewadi are grateful to the Middle East University Amman, Jordan for the financial support granted to cover the publication fee of this research article. Shadi R. Masadeh is grateful to the Isra University, Amman, Jordan for the financial support granted to cover the publication fee of this research article.

REFERENCES

- Al-Husainy, M.A.F. and H.A.A. Al-Sewadi, 2017. A suggested scheme for image cryptography employing genetic algorithm. *J. Theor. Applied Inform. Technol.*, 95: 2619-2626.
- Al-Husainy, M.A.F., 2006. Image encryption using genetic algorithm. *Inform. Technol. J.*, 5: 516-519.

- Alanazi, H.O., B.B Zaidan, A.A. Zaidan, A.H. Jalab, M. Shabbir and Y. Al-Nabhani, 2010. New comparative study between DES, 3DES and AES within nine factors. *J. Comput.*, 2: 152-157.
- Aleisa, N., 2015. A comparison of the 3DES and AES encryption standards. *Intl. J. Secur. Appl.*, 9: 241-246.
- Bhowmik, S. and S. Acharyya, 2011. Image cryptography: The genetic algorithm approach. *Proceedings of the 2011 IEEE International Conference on Computer Science and Automation Engineering (CSAE) Vol. 2*, June 10-12, 2011, IEEE, Shanghai, China, ISBN: 978-1-4244-8727-1, pp: 223-227.
- Bogdanov, A., L.R. Knudsen, G. Leander, C. Paar and A. Poschmann *et al.*, 2007. PRESENT: An ultra-lightweight block cipher. *Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, September 10-13, 2007, Springer, Vienna, Austria, ISBN:978-3-540-74734-5, pp: 450-466.
- Cheng, H., H.M. Heys and C. Wang, 2008. Puffin: A novel compact block cipher targeted to embedded digital systems. *Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD'08)*, September 3-5, 2008, IEEE, Parma, Italy, ISBN:978-0-7695-3277-6, pp: 383-390.
- De Camiere, C., O. Dunkelman and M. Knezevic, 2009. KATAN and KTANTAN-a family of small and efficient hardware-oriented block ciphers. *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'09)*, September 6-9, 2009, Springer, Lausanne, Switzerland, ISBN:978-3-642-04137-2, pp: 272-288.
- Gong, Z., S. Nikova and Y.W. Law, 2011. KLEIN: A new family of lightweight block ciphers. *Proceedings of the 7th International Workshop on Radio Frequency Identification: Security and Privacy Issues (RFIDsec'11)*, June 26-28, 2011, Springer, Amherst, Massachusetts, USA., ISBN:978-3-642-25285-3, pp: 1-18.
- Guo, J., T. Peyrin, A. Poschmann and M.J.B. Robshaw, 2011. The LED Block Cipher. In: *Cryptographic Hardware and Embedded Systems CHES 2011*, Preneel, B. and T. Takagi (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-23950-2, pp: 326-341.
- Hong, D., J. Sung, S. Hong, J. Lim and S. Lee *et al.*, 2006. HIGHT: A new block cipher suitable for low-resource device. *Proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems*, October 10-13, 2006, Yokohama, Japan, pp: 46-59.
- Knudsen, L., G. Leander, A. Poschmann and M.J. Robshaw, 2010. PRINTcipher: A block cipher for IC-printing. *Proceedings of the 12th International Workshop on Cryptographic Hardware and Embedded Systems*, August 17-20, 2010, Springer, Santa Barbara, USA., ISBN: 978-3-642-15030-2, pp: 16-32.
- Kushwaha, P.K., M.P. Singh and P. Kumar, 2014. A survey on lightweight block ciphers. *Intl. J. Comput. Appl.*, 96: 1-7.
- Leander, G., C. Paar, A. Poschmann and K. Schramm, 2007. New lightweight DES variants. *Proc. Int. Workshop Fast Software Encrypt.*, 4593: 196-210.
- Masadeh, S.R. and H.A. Al-Sewadi, 2017. Segmented block cipher algorithm based on ASCII-codes maneuver. *J. Comput. Sci.*, 13: 748-755.
- Schneier, B., 1996. *Applied Cryptography: Protocols, Algorithms and Source Code in C*. 2nd Edn., John Wiley and Sons, New York, USA., ISBN-13: 978-0471117094, pp: 758.
- Shirai, T., K. Shibutani, T. Akishita, S. Moriai and T. Iwata, 2007. The 128-bit blockcipher CLEFIA. *Proceedings of the 14th International Conference on Fast Software Encryption Vol. 4593*, March 26-28, 2007, Springer, Luxembourg, pp: 181-195.
- Stallings, W., 2011. *Cryptography and Network Security: Principles and Practice*. 5th Edn., Prentice Hall, USA., ISBN: 9780136097044, Pages: 719.
- Suzaki, T., K. Minematsu, S. Morioka and E. Kobayashi, 2013. TWINE: A Lightweight Block Cipher for Multiple Platforms. In: *Selected Areas in Cryptography*, Knudsen, L.R. and H. Wu (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-35998-9, pp: 339-354.
- Tsunoo, Y., E. Tsujihara, M. Shigeri, T. Saito and T. Suzaki *et al.*, 2008. Impossible differential cryptanalysis of CLEFIA. *Proceedings of the 15th International Workshop on Fast Software Encryption (FSE'08)*, February 10-13, 2008, Springer, Lausanne, Switzerland, ISBN:978-3-540-71038-7, pp: 398-411.
- Wu, W. and L. Zhang, 2011. LBlock: A lightweight block cipher. *Proceedings of the 9th International Conference on Applied Cryptography and Network Security (ACNS'11)*, June 7-10, 2011, Springer, Nerja, Spain, ISBN:978-3-642-21553-7, pp: 327-344.
- Zhang, W., Z. Bao, D. Lin, V. Rijmen and B. Yang *et al.*, 2014. Rectangle: A bit-slice ultra-lightweight block cipher suitable for multiple platforms. *Master Thesis, International Association for Cryptologic Research (IACR), USA.*