Lightweight Dynamic Crypto Algorithm for Next Internet Generation

Ahmad H. Al-Omari
Computer Science Department, Faculty of Science
Northern Border University, Arar, Saudi Arabia
ahmed.alomari@nbu.edu.sa, kefia@yahoo.com

***Abstract*—Modern applications, especially real time applications, are hungry for high-speed end-to-end transmission which usually conflicts with the necessary requirements of confidential and secure transmission. In this work, a relatively fast, lightweight and attack-resistant crypto algorithm is proposed. The algorithm is a symmetric block cipher that uses a secure pre-shared secret as the first step. Then, a dynamic length key is generated and inserted inside the cipher text. Upon receiving the cipher text, the receiver extracts the key from the received cipher text to decrypt the message. In this algorithm, ciphering and deciphering are mainly based on simple XoR operations followed by substitutions and transpositions in order to add more confusion and diffusion to the algorithm. Experimental results show faster encryption/decryption time when compared to known encryption standards.**

Keywords-dynamic crypto algorithm; lightweight crypto algorithm; dynamic cryptography; shared secrets; next internet generation security

I. INTRODUCTION

Several emerging areas of information and communication technology (ICT) require interconnected devices like Internet of Things (IoT) and sensor networks. IoT and smart applications are growing rapidly and are commonly accessed through smartphones. Currently, more and more smart devices are daily connected to the internet, such as smartphones, smart TVs, video game consoles and even most of the home devices like refrigerators and air-conditioners [1]. All these devices suffer from being resource-constrained regarding their low processing power, limited battery power life, small display size, small memory, and limited storing capacity. As IoT and other smart applications are growing rapidly, they encounter many risks and challenges such as dealing with huge amounts of data, processing power, energy consumption, address security and privacy threats [2]. Security and privacy are fundamental requirements for any application, especially smart applications. The current modern standard cryptographic algorithms were originally designed for traditional desktop/server implementations and many of them consume an unacceptable amount of system resources (computation power, RAM, storage, etc.) and are not suitable for resource-constrained devices.[2]. Therefore, there is a need for lightweight cryptography (LWC) algorithms that suit such resource-constrained devices [3-4].

LWC is one of the most promising research areas in cryptography since it is considered fast in encryption processing, resistant to attacks and low in resource requirements. There are no strict properties needed in order to classify an encryption algorithm as an LWC [5]. According to the National Institute of Standards and Technology (NIST), the main reasons for adopting LWC for smart power constrained devices are the need for efficient end-to-end communication and adoptability in resource-constrained smart devices [3, 6]. Generally, any cryptographic design should take into considerations the tradeoff between security, cost and performance. The performance measurements include power, energy consumption, latency and throughput. Security requirements, on the other hand, aim to maintain an acceptable level of secrecy and privacy of the system. Cryptography, which is part of security, is divided into symmetric and asymmetric cryptography. The symmetric cryptography algorithms use a single private key for encryption and decryption and are originally designed for a wide range of applications that use hardware devices with high processing power and large resources. On the other hand, the asymmetric cryptography algorithms use a pair of keys, a public and a private one. One key is used for encryption and the other for decryption. Traditional symmetric and asymmetric algorithms are not suitable for constrained devices while lightweight cryptographic algorithms are the best choice [7]. Some of the candidate applications for the LWC algorithms include wireless sensor network (WSN), radio-frequency identification, wireless body area network (WBAN), IoT, smart cards, embedded systems, smart systems, etc. [8-9]. These applications support dissimilar devices in heterogeneous environments with minimum human intervention. For example, IoT devices communicate with minimum or no human intervention, a fact that represents a new challenge to the IoT system by both exposing many security attacks as well as gaining unauthorized device access by the attacker device. This may essentially result in severe system damages. Moreover, some IoT implementations are cloud-based applications which have many security issues and challenges [3, 10].

This work focuses on introducing a new model of symmetric block cipher encryption. It is classified as LWC since it requires only a small amount of resources like memory, computing, storage, time and space.

Corresponding author: Ahmad H. Al-Omari

II. RELATED WORK

Lightweight encryption is a recent scientific field. Many lightweight block ciphers (LWBCs) have been proposed. Some of these were modifications and simplifications of traditional block ciphers while others were new like the data encryption standard lightweight (DESL) which is basically based on the original design principles of DES with a variant of using a single S-box instead of eight S-boxes. DESL is claimed to be resistant against most common known attacks like differential, linear, and Davis-Murphy attacks, and it is used in low resource devices like RFID, WSNR, WBN and IoT [11]. Over the last decade, variations of LWC with different properties have been proposed [12]. A word-oriented stream cipher [13] that takes 128-bit as an initial vector and an initial key as inputs while the generated output is a 32-bit key-stream. Afterwards, the key-stream is used to encrypt the plain text. The word-oriented stream cipher algorithm was developed to deal with 8-bit characters in the encryption/decryption process. In each step, the algorithm output is an 8-bit key character, which is bitwise added to the plain-text characters to produce the cipher-text character. The same operation is performed for the decryption process. Theoretically, the proposed algorithm shows high performance through high nonlinear complexity. An extensive literature survey of more than 100 algorithms was performed in order to systemize the concept of LWC in [12]. The survey identified two categories of LWC algorithms. The first is the ultra-LWC, which deals with highly specialized algorithms providing one function with high performance on one platform and the second is the ubiquitous cryptography which deals with multilateral algorithms in terms of functionality and implementation. A new dynamic crypto symmetric algorithm [14-16] that uses a pre-shared secret was proposed to regenerate a predefined table. The regenerated table is again rearranged and shifted many times before the shared-key insertion. The encryption/decryption operations used in this algorithm are simple bitwise XoR operations between the plaintext and the scrambled text. Results show a faster algorithm that achieves better performance than the traditional AES and DES. Moreover, plenty encryption algorithms especially designed for hardware restricted resources can be easily found. For example, MCryption and Crypton are two proposed block ciphers that have the option of using a key size of the length 64-bit, 96-bit or 128-bit. The architecture and the function of each component are simplified in order to run to power-constrained devices [17-18]. Hummingbird-2 is a primitive authentication encryption algorithm especially designed for resource-constrained devices such as RFID tags, WSN and very small hardware or software and it is also suitable for resource-restricted devices. This algorithm uses a 128-bit key and a 64-bit initialization vector [19].

Authors in [21] improved the original work of [20] by enhancing the transformation table composition. It was proved that swapping rows with columns gives better results. Authors in [20] recommended performing the key insertion inside the plain text from both sides simultaneously. Moreover, an enhancement was added to adopt a key size of 128 bytes and plaintext size of 190 bytes. In [21], more enhancements and contributions were added. These enhancements act as adding extra features. The final improvement was using cryptographically secure pseudorandom number generators (CSPRNG) to generate and share the shared value.

III. THE PROPOSED SOLUTION

A. *Background*

This study represents a symmetric encryption algorithm called lightweight dynamic crypto (LWDC) for the next internet generation. The original work of this study first appeared on 2008 [15]. Different researchers added their contributions and enhancements on the original algorithm [14-16, 20-21]. The main architecture of the original algorithm has slightly changed since its first release. The enhancements were added on detailed processes in order to achieve a more stable and attack resistant algorithm. The original algorithm consists of three main processes, the index generation process (IGP), the encryption process (EP) and the decryption process (DP) [16]. The general architecture of the original algorithm is briefly described as:

- The IGP is common between the EP and the DP. First, an initial table and a shared-secret are generated and shared. The shared-secret is used to generate the transformation table (TT) and the table of indexes (TI).
- The EP is performed by XoR-ing the plaintext with the cipher-key to generate the scrambled text. Then the cipher-key is inserted inside the scrambled text according to the values from the TI and the result is the ciphertext (C).
- The DP is performed exactly as the EP process but in the reverse order. In the DP process, the cipher-key is extracted back from the C.

B. *The Solution Architecture*

In this work, additional cryptographic enhancement properties were added to the original algorithm. These enhancements include:

- Using the CSPRNG to generate a random shared secret in order to be difficult, but not impossible, for an adversary to predict.
- Using the CSPRNG to generate a random shared secret key, which should also be difficult, yet not impossible to predict.
- Using the IPsec based on the internet key exchange protocol (IKEv2) to establish a secure connection to exchange data. IPsec is a standard protocol aiming to provide end-to-end security for the internet protocol (IP). The exchanged messages are protected by IPsec and the IPsec session is authenticated using IKEv2 [22].
- Adding the concept of confusion and diffusion (CD) property to the algorithm by implementing substitution and permutation boxes (S-P-(Box)). The CD concept was firstly proposed in [22] as basic building blocks for any cryptographic system. According to [22], the CD concept aims to thwart cryptographic attacks of the statistical cryptanalysis type. Confusion strives to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible while

the diffusion strives to make the statistical relationship between the plaintext and the ciphertext as complex as possible in order to thwart attempts to deduce the key [2, 23].

The three main processes of the algorithm are described briefly below [16]:

1) Index Generation Process

- The shared-value (ShrdV) is randomly generated by using the Blum Blum Shub cryptographically secure pseudorandom number generator (BBS-CSPRNG) and is shared by the IPsec-IKEv2 tunneling protocol.
- The initial table, IniT, is a 16*16 (hex) table, which is fixed and shared between the sender and the receiver.
- The transformation table, TranT, is a table that is generated by performing permutation on the IniT based on the value of ShrdV.
- The indexing table, IndT, is the result of performing another permutation on the TransT based on the value deduced from the TransT.

2) Encryption Process

- The plaintext (P) is the original text that is going to be encrypted.
- The key T is the system key which represents the heart of the encryption process. After generating the key K, it is used to bitwise XoRed P$\oplus$K and is inserted inside the resulted scrambled table ScrT.
- ScrT is the result of performing the XoR operation between the P and the Key.
- The key insertion KeyI is the result of inserting the Key K inside the ScrT.
- The S-BoX is added to the algorithm to enhance the confusion and diffusion properties of the algorithm.
- The ciper text C is the encrypted text.

3) Decryption Process

- The S-BoX is used to recover back the original form of the C before the key recovery (KR) process is performed.
- The KR is the first step in decrypting the C. In the KR process, K is extracted back from the C whereas the ScrT is regenerated back.
- P is recovered back by performing XoR operation between the ScrT and the K, P$\oplus$K.

The LWDC architecture shown on Figure 1 and the pseudo code on Figure 2 represent the general design process architecture of the algorithm. The most important component of any encryption algorithm is the encryption key. The key selection process and the key value should be carefully chosen. The encryption key properties include a key secrecy, a key length and an initial value (seed) [24]. The basic principle in choosing any encryption key (shared value) is to be obtained from one of the known cryptographically secure pseudorandom number generators (CSPRNG) like Lavarand, Simon Cooper or Landon Curt Noll. The strength of CSPRNGs depends on their properties. These properties are represented in the difficulty of finding the next bit to be generated from the previous given sequence of bits without having any clue of the seed in polynomial time. In addition to these properties, the algorithm should satisfy forward and backward unpredictability. All of these properties are found in the Blum Blum Shub (BBS) pseudo random number generator. The BBS is considered as the most preferable algorithm for cryptographic purpose like key generation since it is based on quadratic residue NP-complete problem [25]. Based on that, the BBS-CSPRNG technique was added to generate the shared value ShrdV=f(BBS-CSPRNG) assuming that the IKEv2 protocol is used to share the secret value between the communicating parties.

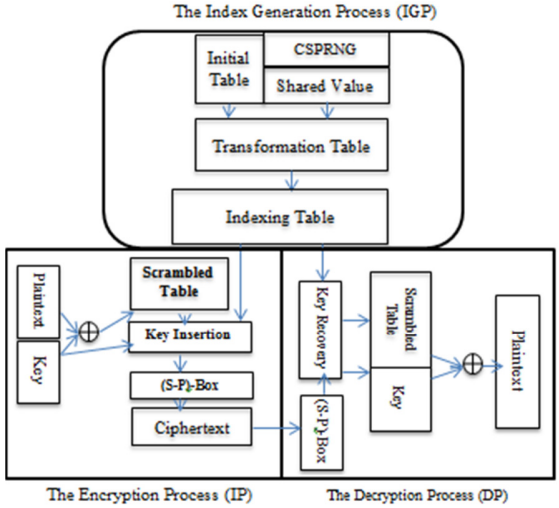

Fig. 1. The lightweight encryption architecture

The Index Generation Process ***IGP***
IPsec establishment
ShrdV = f(BBS-CSPRNG)
TransT = (IniT, ShrdV)
IndxT = f(TransT)
The Encryption Process ***EP***
Scrt = (Plaintxt $\oplus$ Key)
KeyI = f(Scrt, Key, IndxT)
C= S-Box(f(KeyI))
The Decryption Process ***DE***
KR= f((S-BoX(C), IndxT))
ScrT = f(KR)
Key = f(KR)
P= (ScrT $\oplus$ Key)

Fig. 2. The pseudocode

The other enhancement to the algorithm is adding the S-BoX before generating C, the S-BoX will add more CD which is an important property of any block cipher algorithm. CD is performed by applying a constant number of (CD) rounds to

extend the domain of a public random permutation [22]. In the decryption process, the same procedures are performed as in the encryption process but in reverse order. The algorithm could be implemented either in hardware or software. In case of hardware limited resources implementation, it is recommended to burn the algorithm on the hardware chipset while in the software implementation it will be easy to use as a portable and fast encryption-decryption algorithm.

C. *Implementation and Analysis*

The algorithm was tested by using Java JDK 1.7.0_171 and the Java Cryptography Extension (JCE) on a Fujitsu Laptop i7-4702MQ CPU (8-GB RAM, Windows-7). The same files that were used to test the SVSCS algorithm in [21] were used in the experiments and the testing was performed on 10 different file sizes and the results were finally compared with the LWDC results. EP and DP processes were performed on different P sizes. It is worth mentioning that the comparison was performed on the encryption-decryption time which includes the sub-processes of both algorithms (Table I). The key generation process, (S-P)-Box, table scrambling, key insertion, and the encryption time for the different plaintext size are listed on Table I.

TABLE I. ENCRYPTION TIME COMPARISON

		Plaintext Size in MB					
Time	**Alg.**	0.35	0.99	1.65	3.30	6.60	11.80
Key gen.	SVSCS	0.0117	0.0118	0.0112	0.0116	0.0116	0.0118
	LWDC	0.0116	0.0117	0.0110	0.0113	0.0116	0.0118
(S-P)-BoX	SVSCS	0.0058	0.0589	0.1011	0.0502	0.1178	0.3583
	LWDC	0.0059	0.0590	0.1022	0.0502	0.1189	0.3594
Scrambling	SVSCS	0.0286	0.0967	0.1346	0.1987	0.5871	0.7220
	LWDC	0.0272	0.0990	0.1364	0.2004	0.6201	0.7579
Key Insert	SVSCS	0.0132	0.0523	0.0686	0.2590	0.2770	0.6021
	LWDC	0.0145	0.0564	0.0788	0.2675	0.2872	0.6245
Encryption	SVSCS	0.0593	0.2197	0.3165	0.5194	0.9936	1.6942
	LWDC	0.0594	0.2214	0.3180	0.5207	0.9969	1.6994

The encryption time comparison between the SVSCS and our algorithm is shown in Figure 3, in which the encryption time looks equal for both algorithms. In fact, the SVSCS is a little bit faster than our algorithm because the SVCS performs the (S-P)-Box operation before the key insertion process, whereas in our algorithm the (S-P)-Box is performed after the key insertion which expands table size. The rest of the figures (Figures 4-7) show the time variation between the sub encryption operations for both algorithms. The key generation time is shown on Figure 4 where it is clear that our algorithm is a little bit faster than the SVSCS regardless of data size due to the BBS-CSPRNG being used. The (S-P)-Box time which represents the CD properties is shown on Figure 5 where it is clear that our algorithm consumes more time than the SVSCS, due to the (S-P)-BoX operations that are performed on a larger table size than the one of the SVSCS. The table scrambling time is shown in Figure 6 in which no significant time difference is noticed between the algorithms. The key insertion process shown in Figure 7 indicates that there is no significant time difference between the algorithms. Decryption time comparison between SVSCS and our algorithm is listed on Table II. The key generation process is not calculated here since it is generated in the encryption phase.

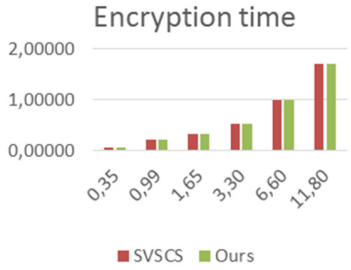

Fig. 3. Encryption time

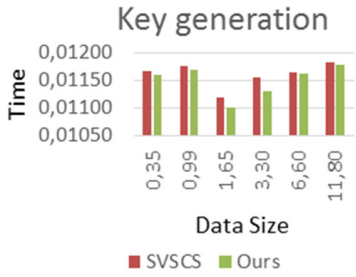

Fig. 4. Key generation time

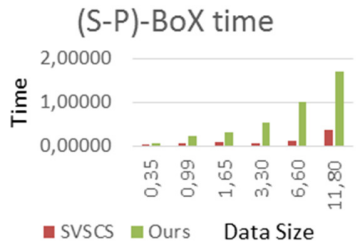

Fig. 5. Confusion and diffusion time

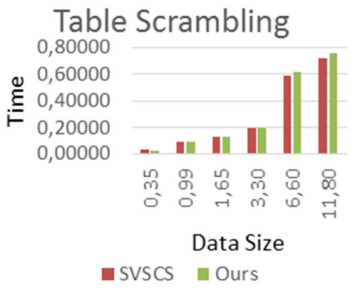

Fig. 6. Table scrambling time

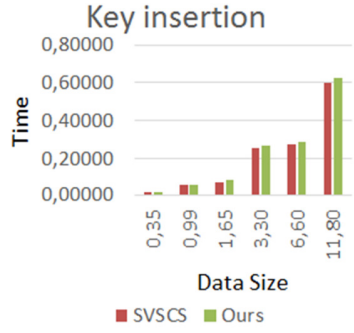

Fig. 7. Key insertion time

Decryption time comparison between the SVSCS and our algorithm is shown in Figure 8. The decryption time looks equal for both algorithms. In some cases our algorithm performs faster than the SVSCS and in other cases the SVSCS performs faster. The time difference is not significant. The algorithm was improved by adding the IPsec-IKEv2 to exchange the secret shared value, by adding the BBS-CSPRNG to generate the secure shared value, and by changing the location of the (S-P)-Box operations in the algorithm. These enhancements give extra randomness (confusion and diffusion) to the cipher text and make it more attack-resistant. However, the added enhancements did not affect the encryption speed negatively, the detailed analysis on [16, 20] is still valid in this enhanced version of the algorithm. On this work, the cipher text becomes more resistant to brute force attacks since the algorithm uses a plaintext size of 190 bytes (1520 bits), key size of 128 bytes (1024 bits) and the (S-P)-BoX. Moreover, using the CD properties in addition to the key insertion process produces a well-mixed and shuffled ciphertext. Thus, it will be hard to solve a plaintext size of (2(1520))×(2(1024)). In this case, the only possibility to attack the algorithm is to use cryptanalysis attacks. From the previous studies, it is proven that the algorithm outperforms the speed of the advanced encryption standard (AES). It is 15 times faster in encryption and 9 times faster in decryption [14-16, 20-21].

TABLE II. DECRYPTION TIME COMPARISON

		Ciphertext Size in MB					
Time	**Alg.**	0.35	0.99	1.65	3.30	6.60	11.80
S-BoX	SVSCS	0.0230	0.0212	0.1455	0.0878	0.1321	0.6987
	LWDC	0.0228	0.0251	0.1634	0.0943	0.1567	0.7012
Scrambling	SVSCS	0.0306	0.1001	0.1532	0.3175	0.6078	0.9874
	LWDC	0.0304	0.1098	0.1612	0.3220	0.6231	0.9913
Recovery	SVSCS	0.0147	0.1182	0.0608	0.1836	0.4827	0.4050
	LWDC	0.0132	0.1163	0.0610	0.1926	0.4931	0.4069
Decryption	SVSCS	0.1215	0.3745	0.6841	1.0050	2.0025	3.7836
	LWDC	0.1117	0.4695	0.5996	1.1099	2.3634	3.4918

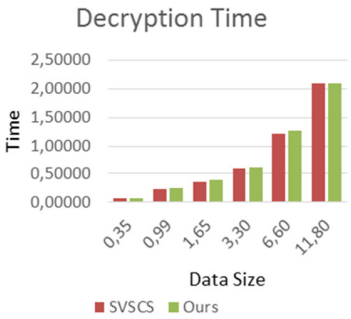

Fig. 8. Decryption time

IV. CONCLUSION

The proposed algorithm shows faster encryption-decryption time than the conventional standard algorithm (AES). The algorithm has the property of hardware and software implementation and hence the uploading of the code on the hardware chipset for faster processing is recommended. The algorithm is simple in nature but very hard to break. Adding the IPsec-IKEv2, the BBS-CSPRNG and the (S-P)-BoX puts the algorithm in the levels of the modern lightweight symmetric encryptions in the market.

ACKNOWLEDGMENT

The authors gratefully acknowledge the approval and the support of this research from the Deanship of Scientific Research study by the grant No. 7189-SCI-2017-1-8-F7, Northern Border University, Arar, Saudi Arabia.

REFERENCES

[1] M. Talbi, F. Maddouri, A. Jemai, M. S. Bouhlel, "Application of a Lightweight Encryption Algorithm to a Quantized Speech Image for Secure IoT", Sixth International Conference on Advances in Computing, Electronics and Communication, Rome, Italy, 2017

[2] S. Singh, P. K. Sharma, S. Y. Moon, J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions", Journal of Ambient Intelligence and Humanized Computing, 2017

[3] K. A. McKay, L. Bassham, M. S. Turan, N. Mouha, Report on Lightweight Cryptography. Technical Report NISTIR 8114, National Institue of Standards and Technology, 2017

[4] G. Bansod, N. Raval, N. Pisharoty, "Implementation of a new lightweight encryption design for embedded security", IEEE Transactions on Information Forensics and Security, Vol. 10, No. 1, pp. 142-151, 2015

[5] B. Chaitra, V. G. K. Kumar, R. C. Shatharama, "A Survey on Various Lightweight Cryptographic Algorithms on FPGA", IOSR Journal of Electronics and Communication Engineering, Vol. 12, No. 1, pp. 45-59, 2017

[6] K. P. Mahaffey, J. G. Hering, J. D. Burgess, J. P. Grubb, D. Golombek, D. L. Richardson, A. McKay Lineberry, T. M. Wyatt, System and Method for Mobile Communication Device Application Advisement, U.S. Patent No. 9,367,680, 2016

[7] Isha, A. K. Luhach, "Analysis of Lightweight Cryptographic Solutions for Internet of Things", Indian Journal of Science and Technology, Vol. 9, No. 28, 2016

[8] W. Joseph, B. Braem, E. Reusens, B. Latre, L. Martens, I. Moerman, C. Blondia, "Design of energy efficient topologies for wireless on-body channel", 17th European Wireless 2011-Sustainable Wireless Technologies, Vienna, Austria, April 27-29, 2011

[9] J. Yick, B. Mukherjee, D. Ghosal, "Wireless sensor network survey", Computer Networks, Vol. 52, No. 12, pp. 2292-2330, 2008

[10] A. Sajid, H. Abbas, K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges", IEEE Access, Vol. 4, pp. 1375-1384, 2016

[11] A. Poschmann, G. Leander, K. Schramm, C. Paar, "New Light-Weight Crypto Algorithms for RFID", IEEE International Symposium on Circuits and Systems, New Orleans, USA, May 27-30, 2007

[12] A. Biryukov, L. P. Perrin, "State of the art in lightweight symmetric cryptography", available at: http://orbilu.uni.lu/handle/10993/31319, 2017

[13] H. M. S. El Hennawy, A. E. A. Omar, S. M. A. Kholaif, "LEA: link encryption algorithm proposed stream cipher algorithm", Ain Shams Engineering Journal, Vol. 6, No. 1, pp. 57-65, 2015

[14] A. H. Omari, B. M. Al-Kasasbeh, A. A. Omari, "Dynamic Cryptography Algorithm for Real-Time Applications DCA-RTA", 3rd International Conference on Applied Mathematics, Simulation, Modelling, Circuits, Systems and Signals, Athens, Greece, December 29-31, 2009

[15] A. H. Omari, B. M. Al-Kasasbeh, R. E. AL-Qutaish, M. I. Muhairat, "New Cryptographic Algorithm for the Real Time Applications", 7th WSEAS International Conference on Information Security and Privacy, Cairo, Egypt, December 29-31, 2008

[16] A. H. Al-Omari, "Dynamic Crypto Algorithm for Real-Time Applications DCA-RTA, Key Shifting", International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, pp. 72-77, 2016

[17] C. H. Lim, T. Korkishko, "mCrypton–A lightweight block cipher for security of low-cost rfid tags and sensors", in: Information Security

Applications, Lecture Notes in Computer Science, Vol. 3786, pp. 243–258, Springer, 2005

[18] C. H. Lim, Crypton: A New 128-bit Block Cipher, NIsT AEs Proposal 1998

[19] D. Engels, M. J. O. Saarinen, P. Schweitzer, E. M. Smith, "The Hummingbird-2 lightweight authenticated encryption algorithm", 7th International Workshop on Security and Privacy, Amherst, USA, June 26–28, 2011

[20] A. A. Al-Omari, Investigating a Dynamic Crypto Algorithm for Real Time Applications (DCA-RTA), MSc Thesis, The University of Jordan, 2012

[21] M. A. Al-Qaysi, A Shared Value Based Symmetric Crypto System (SVSCS), MSc Thesis, Princess Sumaya University for Technology, 2014

[22] C. Cremers, "Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2", in: European Symposium on Research in Computer Security, Lecture Notes in Computer Science, Vol. 6879, pp. 315-334, Springer, 2011

[23] Y. Dodis, M. Stam, J. Steinberger, T. Liu, "Indifferentiability of Confusion-Diffusion Networks", in: Advances in Cryptology–Eurocrypt 2016, Lecture Notes in Computer Science, Vol. 9666, pp. 679-704, Springer, 2016

[24] H. Feistel, Cryptographic coding for data-bank privacy, IBM Thomas J. Watson Research Center, 1970

[25] G. Singh, Supriya, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security", International Journal of Computer Applications, Vol. 67, No. 19, pp. 33–38, 2013

[26] Divyanjali, Ankur, V. Pareek, "An overview of cryptographically secure pseudorandom number generators and BBS", International Conference on Advances in Computer Engineering & Applications, Ghaziabad, India, February 15, 2015