An Effective Lightweight Cryptographic Algorithm to Secure Resource-Constrained Devices

Sohel Rana[1]
Department of CSE
Bangladesh University of Business & Technology,
Mirpur-2, Dhaka,
Bangladesh

Saddam Hossain[2], Hasan Imam Shoun[3], Dr. Mohammod Abul Kashem[4]
Department of CSE
Dhaka University of Engineering & Technology, Gazipur
Gazipur, Bangladesh

Abstract—In recent years, small computing devices like embedded devices, wireless sensors, RFID tags (Radio Frequency Identification), Internet of Things (IoT) devices are increasing rapidly. They are expected to generate massive amount of sensitive data for controlling and monitoring purposes. But their resources and capabilities are limited. Those also work with valuable private data thus making security of those devices of paramount importance. Therefore, a secure encryption algorithm should be there to protect those vulnerable devices. Conventional encryption ciphers like RSA or AES are computationally expensive; require large memory but hinder performances of those devices. Simple encryption techniques, on the other hand are easy to crack, compromising security. In this paper a secure and efficient lightweight cryptographic algorithm for small computing devices has been proposed. It is a symmetric key block cipher, employing custom substitution-permutation (SP) network and a modified Feistel architecture. Two basic concepts from Genetic algorithm are used. A Linux based benchmark tool, FELICS is used for the measurement and MATLAB for the purpose of encryption quality testing. An improvement over the existing algorithm, the proposed algorithm reduces the use of processing cycles but at the same time provides sufficient security.

Keywords—Lightweight cryptography; IoT; RFID tags; genetic algorithm; feistel architecture; SP network; FELICS; MATLAB

I. INTRODUCTION

Lightweight cryptography [1] is a sub-category in the field of cryptography that intends to provide security solutions for resource-constrained devices. Cryptography means "secret writing" [2]. In computer communication all want to encrypt information so that no unwanted entity but the expected one can decipher the information. At the core of lightweight cryptography there is a trade-off between security and lightweightness: that is how anyone can achieve a good level of security in small computing devices? Recently, academic communities have been doing a significant amount of work related to lightweight cryptography; to implement conventional cryptography standards efficiently, to design and analyze new lightweight algorithms and protocol. The widespread utilization of small computing devices such as sensors nodes, Radio-Frequency Identification (RFID) tags, industrial controllers and smart cards indicates there have been massive changes in people's lives. New security and privacy considerations arise as one shift from desktop computer to small devices. It is challenging to implement heavyweight cryptographic standards to small devices [3]. Many conventional cryptographic algorithms, was optimized for desktop and server environments. Optimization in terms of security, performance and resource requirements makes those algorithms difficult or impossible to implement in resource-constrained devices. Even if they can be implemented, they hinder the performance on the small devices. Lightweight cryptography aims at wide variety of hardware and software spectrum in which an algorithm can be implemented. On the device spectrum in Figure 1 for example, servers and desktop computers occupy at the high end [1]. Tablets and smartphones are the next.

Servers and Desktops	Conventional cryptography
Tablets and Smartphones	
Embedded Systems	Lightweight cryptography
RFID and Sensor Networks	

Fig. 1. Device Spectrum.

Conventional cryptographic algorithms inherently perform well in these devices. Embedded systems, RFID devices and sensors networks can be found at the end in the spectrum. Highly resource-constrained devices are at the very end of the spectrum that has very limited processing capabilities and memory. Lightweight cryptography is principally motivated for those.

Microcontrollers of wide array of performance traits are available. 8-bit, 16-bit and 32-bit microcontrollers are more common but use of 4-bit microcontrollers for certain ultra-low cost applications are noticeable. There exist some instruction sets which only contain a small number of simple instructions. When executing common cryptographic algorithm, they take excess number of cycles. The intended application can get slower and energy-consuming. The amount of random-access memory (RAM) and read-only memory (ROM) of certain microcontroller can be very limited; ranging from 64 bytes to as little as 16 bytes. RFID and sensors are often used in applications which require very strict timing and power requirements [3]. They are for only dedicated purpose and their constraints are stringent. The algorithm they need must also fulfill their requirements.

It is important to understand that lightweight cryptography is not necessarily only for the lower end devices of the

spectrum. Many resource-constrained devices work with server which is powerful. The server must support lightweight algorithm so that it can interoperate with the devices.

A. Motivation

Cryptography itself a challenging and interesting subject to study and especially to research on. It is impossible to think secure data communication without cryptography. In history, people won wars using cryptography as a weapon. It involves mathematics, algorithm, programming, understanding in data communication, etc. With the widespread use of small low powered devices, lightweight cryptography will play a vital role in future. A survey by HP states that more than 70% of resource-constrained devices are vulnerable [4]. It is necessary to make a balance between the security and performance.

B. Security Challenges in Resource-Constrained Devices

Resource-constrained devices have many application areas: automotive systems, smart parking, sensor networks, disaster/weather forecast, healthcare, distributive control systems, Internet of Things (IoT), cyber-physical systems, smart cities, smart grid, etc. Building the confidence among the user is necessary for the adoption of technology with small computing devices, especially about its privacy and security [5]. Resources constrained devices are intrinsically defenseless to many types of security threats.

Devices in IoT are extremely open to assaults [6]. As they remain unsupervised for long time there is a chance of physical attack on its components. Also eavesdropping is simple because of wireless communication medium. The constituents bear low competency in terms of energy and computational capability. If conventional security algorithms are used which require computations, their performance will be wasted [7]. IoT, used for monitoring purposes generates substantial amount of data, so their integrity and authentication are a matters of concerns.

The confidentiality of the data is retained in secure systems. It is important that data should retain its originality and no intentional or inadvertent changes are undetected by the system [8]. For example IoT is composed of many small devices such as RFIDs which remain unattended for a long time [9]; it is easier for any malicious entity to steal the data stored in the memory.

II. Literature Review

In [10], authors enhance the security of Ceaser cipher including sharing secret key using modified Diffie-Hellman technique. Shared key are made in the following way: Let A has a public key 10 and private key 14. A sends 140 (public key multiplied by private key) to B over unsecure channel. B also has a private key 16, so B sends 160 to A. A generate the value of shared secret key as 140 multiplied with 16 result is 2240. Similarly B generates the key value 160 multiplied with 14 which is same to 2240. They use the mod operation with 26 to get the value less than or equal to 26. For any character in the 'x' position the secret key is simply first multiplied with 'x' and then mod is done to get the cipher character. So 2nd character of the message is multiplied with 2, third character with 3 and so on. Then some light calculation to perform cipher.

Authors in paper [11] analyze the performance and security of different type of lightweight encryption algorithm, which are used in especially resource-constrained applications. Four lightweight algorithms TEA, HIGHT, KATAN and KLEIN are implemented on AVR Atmel ATtiny45 microcontroller to evaluate performance analysis on their memory efficiency and energy consumption and also evaluated degree of confusion and diffusion for security analysis.

In paper [12], the authors propose an encryption technique using simple mathematical operations and trivial authentication using unique id. The algorithm applies encryption on ASCII values. Each receiver has unique id and sender possesses a database of all receivers. A set of three keys are used. First a palindrome number is generated from receiver's alphanumeric id and four random numbers. From the palindrome number an encoding matrix is generated. Data is encrypted using the encoding matrix and ASCII values of data. The decryption process is done using the inverse of the encoding matrix called the decoding matrix. But the entire process is questionable to security analysis. Here the encrypted data, keys and random number seed are sent to the receiver. It is possible for any intruder to perform a middle-man attack thus making the entire process vulnerable.

In [13], the Authors propose an algorithm based on combined concept of Genetic Algorithm (GA) and pseudorandom number sequence generation. Only two operators (Crossover and Mutation) of GA are used as a part of this algorithm. They used Blum Blum Shub to generate the pseudorandom sequences in order to select the crossover operators among three (single point, two points, and uniform crossover). Also, five keys are used for performing the encryption and decryption process. First key is a number that indicates a size of block to divide the plain text into blocks. Second and third keys are used to generate the random sequences. Fourth key indicates the modulating factor and Fifth key is used for mutation oparation. This algorithm ensures higher performance and security through the concept of GA and pseudorandom sequence generation.

In [14] The authors proposes a symmetric key block cipher that uses 64 bit key over 64 bit data. Block ciphers such as AES uses substitution-permutation (SP) network in order to integrate Shannon's confusion and diffusion properties. Other ciphers such as Blowfish and DES use Feistel architecture using the advantage of having almost the same encryption and decryption operation.

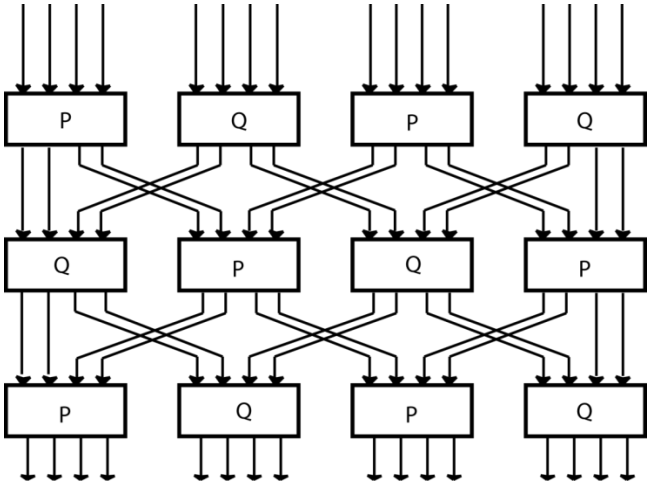


Fig. 2. F-Function of SIT algorithm.

Their proposal is a combination of both Feistel and SP networks (here F-function as SP network as in figure 2), using properties of the both to provide substantial security but keeping the computation complexities as minimum as possible. The algorithm has two parts: key expansion and encryption. A 64 bit key is input by user, divided into 4 blocks, supplied into F-functions, arranged in 4X4 matrices and new five unique keys are generated using some linear and non-linear transformations.

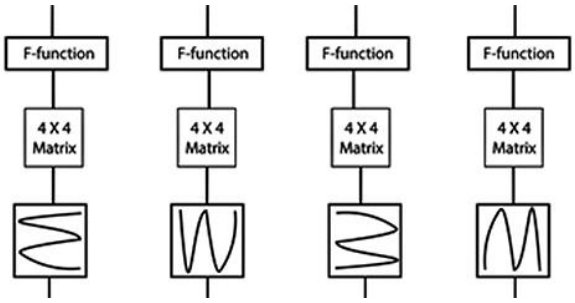


Fig. 3. 4x4 matrices formation during key scheduling.

Input from the f-function forms a matrix and a non-linear transformation occurs as show in figure 3. It can be observed that it is too time consuming and with a little tweak the operation can be minimized using in-place bit shuffling and introducing random number.

The encryption process consists of logical operations, shifting, and substitutions. Although other cipher uses 10 to 20 rounds but it uses Feistel network of 5 rounds that use the five unique generated keys but provides enough confusion and diffusion.

III. Proposed Algorithm

The proposed algorithm is a symmetric key block cipher. It constitutes 64-bit key. In any symmetric key algorithm the encryption process is made up of several encryption rounds. Some mathematical functions define each round to create confusion and diffusion. Increasing number of rounds will ensure better security but will increase the consumption of the device. A typical cryptographic algorithm usually consists of on average 10 to 20 rounds so that the encryption process is strong enough. But the proposed algorithm restricted to only five rounds. The algorithm utilizes the Feistel network. It creates sufficient confusion and diffusion of data so that attacks can be confronted.

The algorithm consists of two parts:

a) Key Scheduling

b) Encryption Process

Key is the most fundamental component in the process of encryption and decryption. The entire security of the data is dependent on the key. The secrecy of the data will be lost if an attacker happens to know the key. Therefore, the revelation of the key should be as difficult as possible. The Feistel network used here consists of five rounds each requiring five unique keys for the encryption/decryption purpose. On figure 4 the key scheduling block is illustrated.

The proposed algorithm requires a 64-bit key. A 64-bit of data can be encrypted or decrypted using that key. In order to guard against exhaustive search attack, the length of the first key must be large enough so that it becomes difficult for the enemy to perform key searching attacks. A cipher key is taken as an input which is 64-bit. The cipher key is input to the key expansion architecture. The block creates five unique keys after going through much confusion and diffusion. The modification that is made from the existing algorithm is shown in the dashed border. Inside the border there are four blocks called non-linear bit shuffling replacing conventional matrix operation. The non-linear bit-shuffling is efficient in creating more confusion and diffusion than the other non-linear operation.

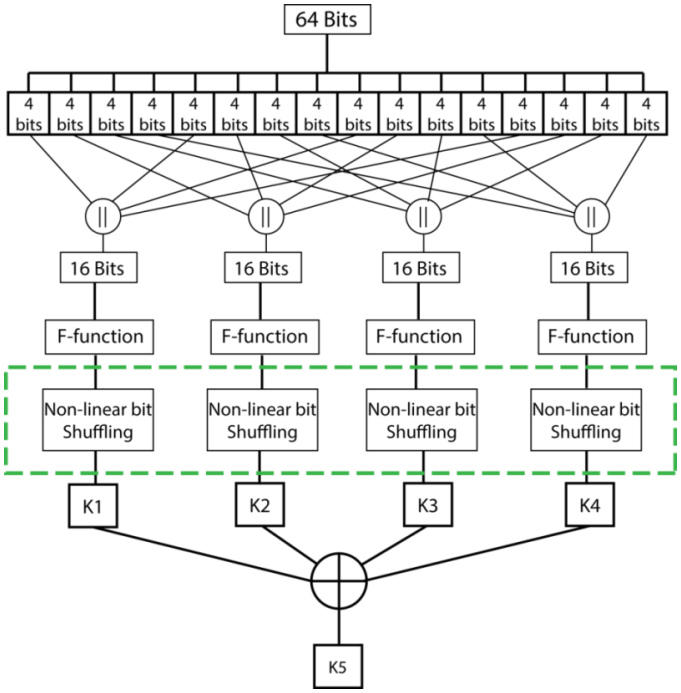


Fig. 4. Key Scheduling

The non-linear bit-shuffling block replaces matrix operation in existing method. A 16-bit input from F-function enters into the block. Taking that 16-bit data as a seed a

pseudo random number is generated using linear feedback shift register. Then the two is XORed. The result is transferred to the bit shuffling block that performs a operation as shown in figure 6. The bit shuffling blocks perform an in-place conventional permutation. Figure 5 illustrates how these operations are performed.

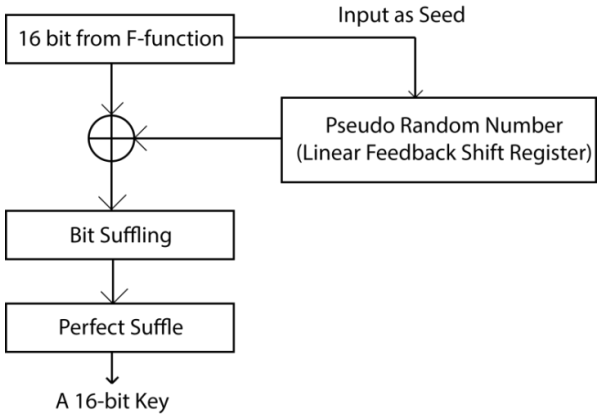


Fig. 5. Process of non-linear bit shuffling block.

Again the output enters into the perfect shuffling block. Figure 7 illustrates how a perfect shuffle is performed.

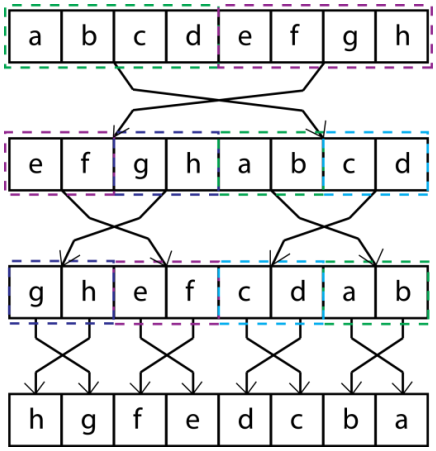


Fig. 6. Bit shuffling.

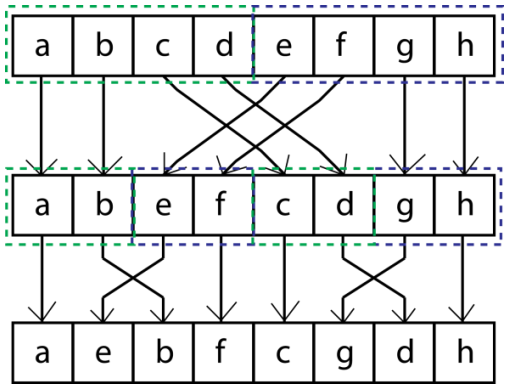


Fig. 7. Perfect shuffle

First four keys, K1, K2, K3, K4 are generated after non-linear bit shuffling. The fifth key K5 is computed by the XOR of the keys K1-K4.

The encryption process encrypts a 64-bit block of data in five rounds using five unique keys generated in the key expansion block. To create considerable confusion and diffusion this process is composed of some shifting, swapping, substitution, XOR, XNOR operations.

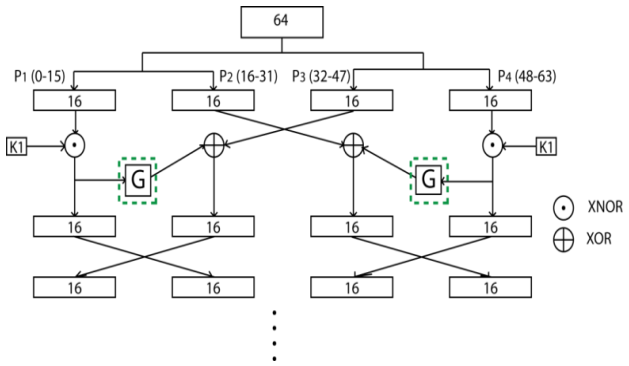


Fig. 8. One of the rounds encryption process.

For the first round an array (figure 8) of 64 bit plain text is first divided into four segments of 16 bits Px_{0-15}, Px_{16-31}, Px_{32-47}, and Px_{48-63}. As the bits progresses in each round the swapping operation is applied so as to diminish the data originality by altering the order of bits, essentially increasing confusion in cipher text. Bitwise XNOR operation is performed between the respective round key K_i obtained earlier from key expansion process. The output of XNOR operation is fed to the modified G-function. The rounds are repeated using the following equations.

$$Ro_{i,j} = \begin{cases} Px_{i,j} \odot K_i \quad ; & j = 1 \; and \; 4 \\ Px_{i,j+1} \oplus EG_{li} \quad ; & j = 2 \\ Px_{i,j-1} \oplus EG_{ri} \quad ; & j = 3 \end{cases} \tag{1}$$

The results of the final round are concatenated to obtain Cipher Text (Ct).

The encryption process consists of five rounds and uses Feistel architecture. The data block is of 64-bit. The 64-bit data is divided into four 16-bit data. Each round utilizes one key; first round uses first key, second round uses second key and so on. Each key is used twice. In each round the innovated G-function is also used twice. This considerably reduces processing cycles. Figure 9 shows how five rounds of encryption looks like. Please note, after each round data blocks are exchanged except the last round. The decryption process is the opposite of the encryption process. This time last key is used first.

A. G-Function

Two fundamental concepts from genetic algorithm called crossover and mutation are used in the function. That is why the function is named as G-function. Figure 10 illustrates the process of a G-function. The block takes a 16-bit input. The input is divided into two 8-bit blocks. Middle four bits are substituted using a substitution box which is precomputed inside the program. Then a two-point crossover is performed over the two 8-bit blocks. Then a simple mutation is performed. It uses coin flip operation. In coin flip mutation only the first bit is flipped, that is 1 flipped to 0 and 0 is flipped to 1.

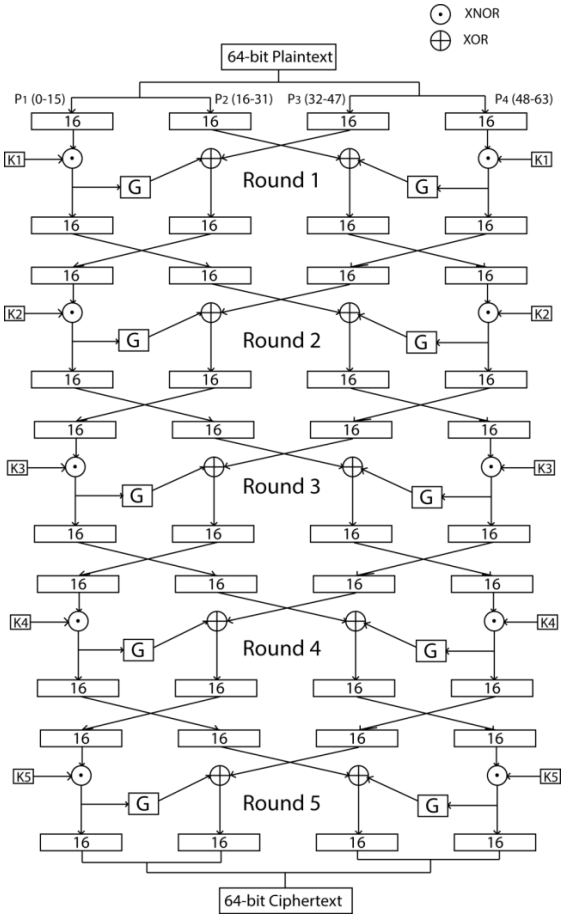


Fig. 9. Five rounds of the encryption process.

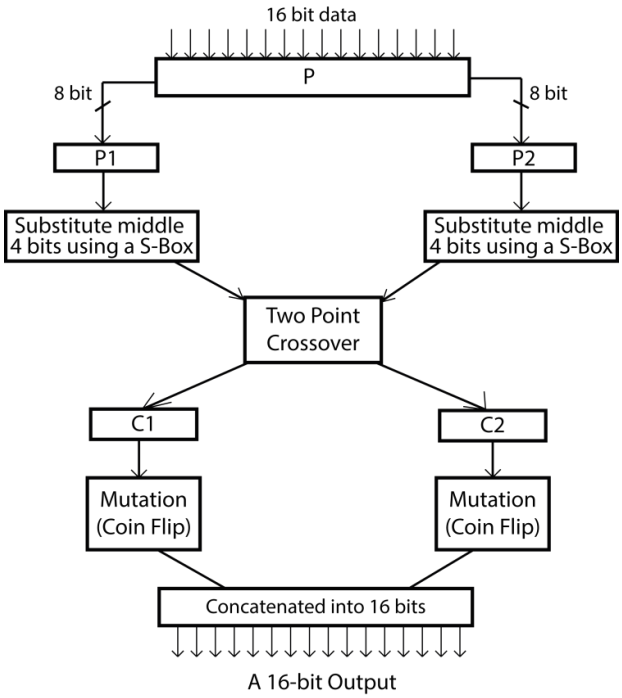


Fig. 10. Process of the G-function.

IV. Experimental Setup

The proposed algorithm is implemented in C programming language. C is an excellent choice for low-level operation as cryptographic algorithms require bit level operations. The algorithm is coded using Visual C++ Express 2010, although CodeBlocks is an excellent choice. The coding was independent of any machine specification. In order to measure execution cycles, memory usage a fantastic benchmark tool called FELICS (Fair Evaluation of Lightweight Cryptographic Systems) [15] is used. This tool incorporates already other standard and popular lightweight cryptographic algorithms like AES, PRESENT, HIGHT, SIMON, SPECK, among others. Many of these are implemented in different version optimized for different consideration in mind. FELICS provides interface to facilitate implementation of any new algorithm and comparing with standard ones. The tool is available to be downloaded. It runs on Linux Ubuntu. A virtual machine file incorporates both Linux Ubuntu and FELICS that saves the user from installing all prerequisites. In the experiment, the virtual machine file is used and that works excellently. The proposed algorithm is also implemented in MATLAB in order to analyze security strength by encrypting images.

The security strength of proposed algorithm is tested to evaluate the basis of following criterion: Key sensitivity, change of cipher entropy, histogram and correlation of the image. Main considerations for observation are the memory utilization and execution cycles for key generation, encryption and decryption of this algorithm.

A. Key Sensitivity

Key sensitivity ensures that the cipher must not decipher to original data if the key has even a bit difference from the actual key. The amount of change occurred in the ciphertext by the change of one bit of the key is evaluated by Avalanche test. According to Strict Avalanche Criterion, the test is to be perfect if 50% of the bits are changed effect of one bit change [16]. To practically observe this effect, an image is decrypted with a key which has only one bit difference from the actual key.

B. Execution Cycle

Most fundamental parameter for the evaluation of algorithm performance is the amount of cycle to perform encoding and decoding a particular data. The proposed algorithm developed for resource-constraint devices in mind must consume minimal cycle and should offer desired security. Execution cycle and power consumption can be correlated, in which case minimizing the cycle also tends to reduce the power consumption.

C. Memory Utilization

Limitation of memory is one of the major challenges for resource-constraint devices. Memory can be measured the number of registers and the number of bytes of RAM and ROM that are used. ROM is used to store the program code and fixed data such as S-boxes and hardcore round keys, while RAM is used to store the computational values. The proposed algorithm uses small amount of rounds that suitable and favorable for its deployments in resource-constraint devices.

D. Histogram

The image histogram is a powerful technique to observe the strength of security for a particular cryptographic algorithm. It is also a basic tool for quality control. However, a histogram can measure the randomness while encrypting an image. A cryptographic algorithm refers to enough secure if the calculated histogram after encryption is uniform.

E. Image Entropy

Image entropy is a quantity which is used to describe the amount of data that must be coded by an encryption algorithm. Higher entropy of an image after encryption refers to the higher security of encryption algorithm. An 8 bits gray scale image can have maximum entropy of 8 bits. Image entropy can be calculated using following equation as below.

$$Entropy,\ \ H(I) = -\sum_{i=1}^{2^8} P(I_i)\log_2 P(I_i) \quad \text{.....(2)}$$

Where $\boldsymbol{P(I_i)}$ is the probability that the difference between 2 adjacent pixels is equal to i.

F. Correlation

The Correlation is an effective way to measure the strength of a cryptography algorithm. However, the correlation between two values refers to the dependency. The cipher text of corresponding plaintext has no dependency on its original data or plaintext for an ideal block cipher. Hence, no information can be uncovered from the cipher text only [17]. Correlation coefficients for original and encrypted messages are calculated using the following equation.

$$\Upsilon_{x,y} = \frac{cov(x,y)}{\sqrt{D(x)\sqrt{D(y)}}} \quad \text{.....(3)}$$

Where $\boldsymbol{cov\,(x,y)}$, $\boldsymbol{D(x)}$ and $\boldsymbol{D(y)}$ are covariance and variances of variable *x* and *y* respectively. Also, $\boldsymbol{D(\alpha)}$ and $\boldsymbol{cov\,(x,y)}$ can be calculated as follows,

$$D(\alpha) = \frac{1}{N}\sum_{i=1}^{N}(\alpha_i - E(\alpha))^2 \quad \text{.....(4)}$$

$$cov\,(x,y) = \frac{1}{N}\sum_{i=1}^{N}\big(x_i - E(x)\big) * \big(y_i - E(y)\big) \quad \text{.....(5)}$$

Where $\boldsymbol{E(x)}$and $\boldsymbol{E(y)}$ are the expected values of variable x and y .Also $\boldsymbol{E(\alpha)}$ can be evaluated using the following equation.

$$E(\alpha) = \frac{1}{N}\sum_{i=1}^{N}\alpha_i \quad \text{.....(6)}$$

Where N is the total number of pixels of the image, $\boldsymbol{\alpha_i}$ is a vector of length N and $\boldsymbol{\alpha_i}$ is the ith intensity values of the original image.

FELICES provides an command line interface like gcc (GNU Complier Collection) to test and build any lightweight cryptographic code. They have already provided standard and popular lightweight cipher. In order to test any other cipher the algorithm should be coded in specific format. They've provided documentation to facilitate the implementation. Anyone can compile his implementation and tests whether the algorithm is runnable in FELICS or not. It provides three scenarios against which one can test his code. It is a very convenient and highly advisable tool. Figure 11 illustrates an example of a run.

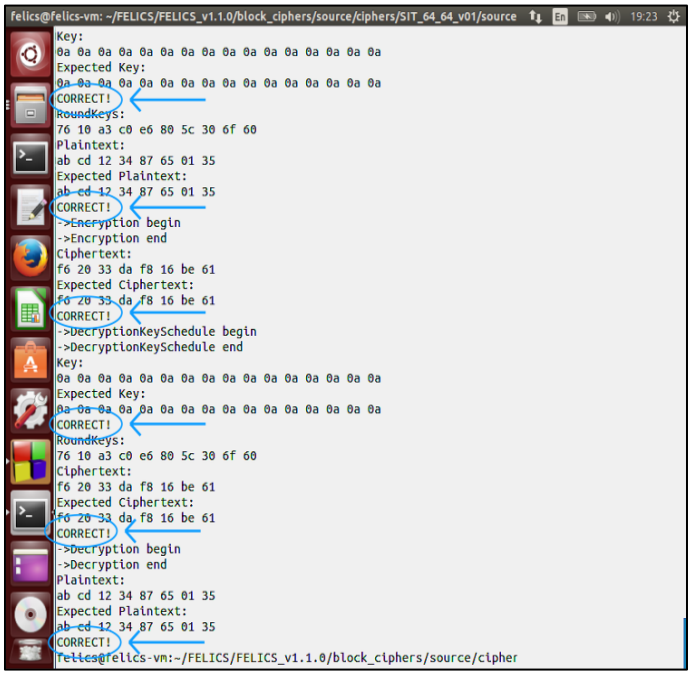


Fig. 11. Testing the implementation of the proposed algorithm on FELICS.

TABLE I. DATA TABLE FOR DIFFERENT CIPHERS IMPLEMENTED ON AVR ARCHITECTURE.

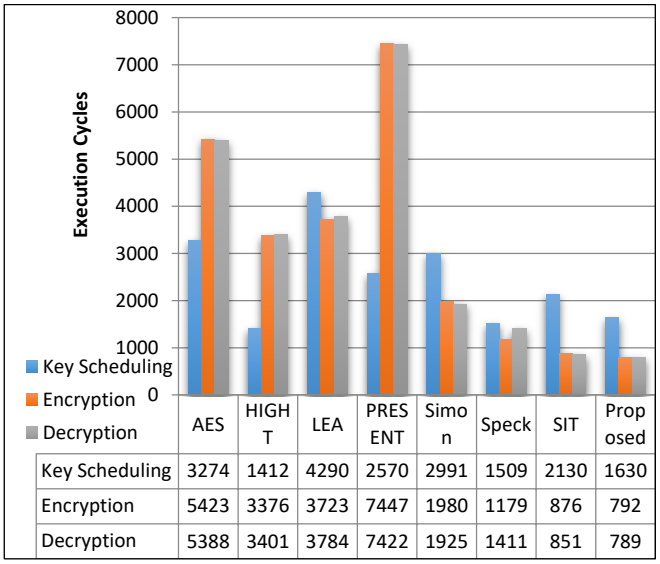
CIPHER	*DEVICE*	*Block Size*	*Key Size*	*Code Size*	*RAM*	*Cycles (key generation)*	*Cycles (encryption)*	*Cycles (decryption)*
AES	AVR	128	128	23090	720	3274	5423	5388
HIGHT	AVR	64	128	13476	288	1412	3376	3401
LEA	AVR	128	128	3700	432	4290	3723	3784
PRESENT	AVR	64	80	1738	274	2570	7447	7422
RC5	AVR	64	128	20044	360	26793	4616	4652
Simon	AVR	64	96	1370	188	2991	1980	1925
Speck	AVR	64	96	2552	124	1509	1179	1411
SIT	AVR	64	64	826	22	2130	876	851
PROPOSED	**AVR**	**64**	**64**	**1228**	**34**	**1630**	**792**	**789**

The simulation of the algorithm is performed by popular open source benchmark tool for lightweight cryptography

FELICS (Fair Evaluation of Lightweight Cryptographic Systems). It uses different platforms (such as AVR, MSP, ARM and PC) for performance evaluation and usually in different conditions. It can also evaluate execution cycles, RAM footprint and binary code size. It can easily compare new cipher with previous one.

Results comparison of different Lightweight Algorithm for Hardware Implementation are shown in TABLE I

In the bar chart in figure 12, comparisons are illustrated among different lightweight algorithm along with the proposed algorithm. The comparisons are made based on number of cycles taken by key scheduling, encryption and decryption individually. The chart shows that the proposed algorithm executes in fewer number of cycles, significantly improving over the others.



	AES	HIGHT	LEA	PRESENT	Simon	Speck	SIT	Proposed
Key Scheduling	3274	1412	4290	2570	2991	1509	2130	1630
Encryption	5423	3376	3723	7447	1980	1179	876	792
Decryption	5388	3401	3784	7422	1925	1411	851	789

Fig. 12. Execution Cycle Comparison for Hardware Implementation.

Two plots in figure 13 and figure 14 demonstrates encryption and decryption cycles of data size between 64 bits and 1024 bits. The proposed algorithm can be seen as a green line taking fewer cycles.

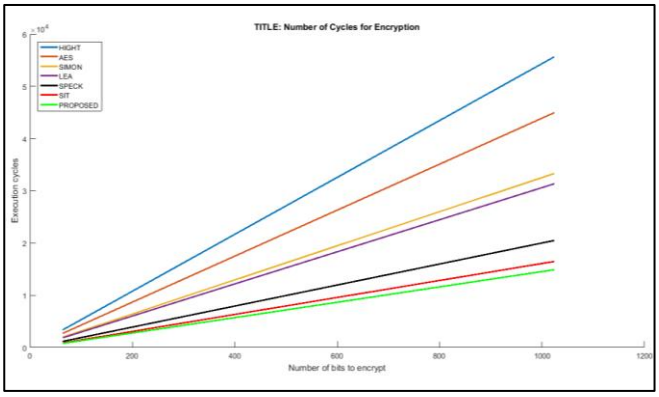


Fig. 13. Execution cycle curve for different cipher in different data sizes for encryption.

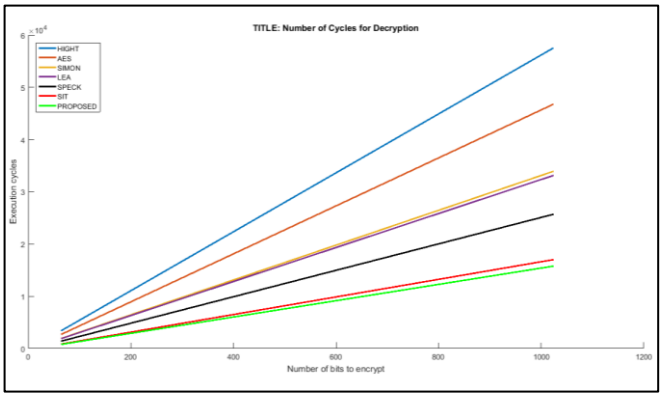


Fig. 14. Execution cycle curve for different cipher in different data sizes for decryption

For a visual observation of encryption-decryption demonstrate the code in MATLAB® which decrypted data using correct key.

The avalanche test of the algorithm, as in figure 15, implies that a single bit change in key, the plaintext brings around 49% changes in cipher bits. The decryption is non-recognizable if even one bit changed in original keys.

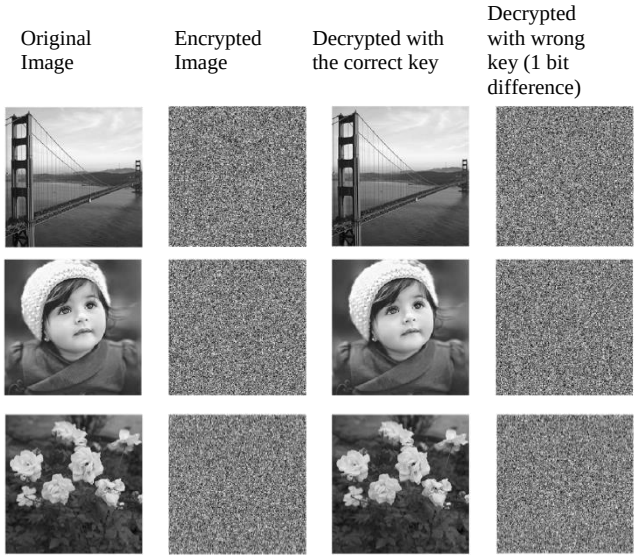


Fig. 15. Analysis of Key Sensitivity

In figures 16, 17, 18, the vertical lines indicate the number of pixels and the horizontal lines indicate the intensity value for each histogram. After encryption, uniform distribution of intensities indicates desired security.

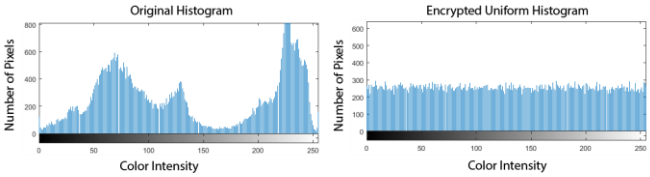


Fig. 16. Bridge histogram

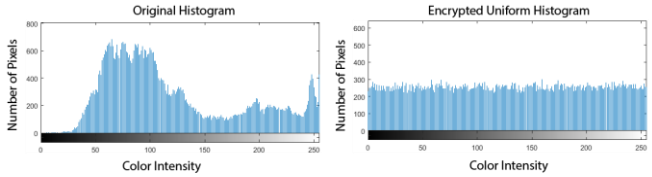


Fig. 17. Child histogram

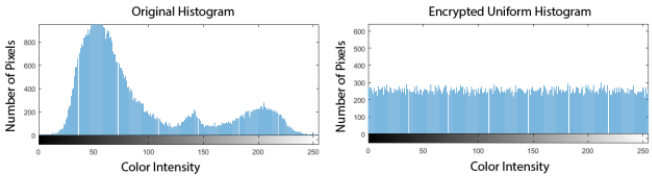


Fig. 18. Flower histogram

The correlation graphs in figures 19, 20, 21 demonstrate the comparison between the original images and the encrypted images. The original image demonstrates highly correlated value whereas the encrypted image seems to have negligible correlated value. Less correlation gives better security for the intended purpose.

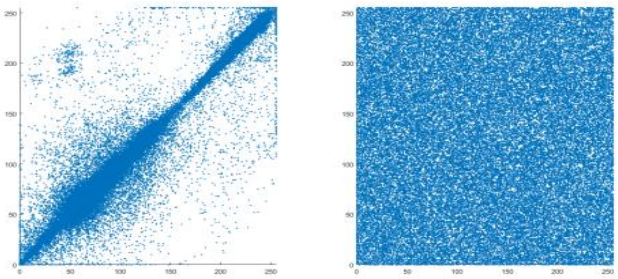


Fig. 19. Bridge correlations for encrypted and decrypted image.

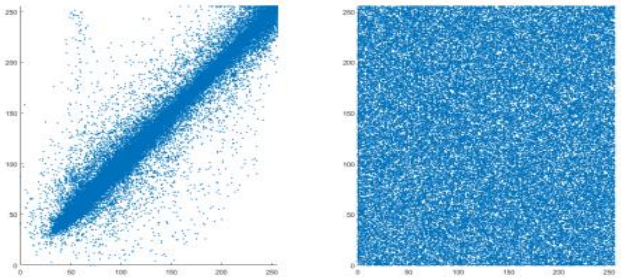


Fig. 20. Child correlations for encrypted and decrypted image.

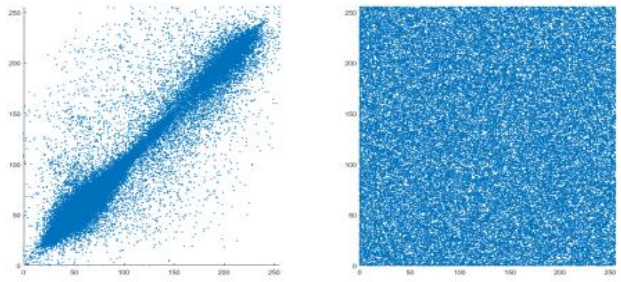
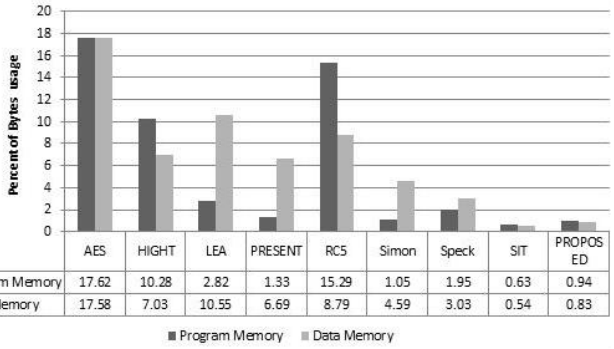


Fig. 21. Flower correlations for encrypted and decrypted image.

The performance of lightweight algorithms in term of memory efficiency is analyzed based on the size of the SRAM and In-System Programmable Flash. Figure 22 compares the memory usages with the existing algorithm. The size of SRAM and In-System Programmable Flash for Atmel ATmega128 microcontroller is 4KB and 128k bytes respectively. The program memory usage bases on the size of Assembly code for each algorithm.



	AES	HIGHT	LEA	PRESENT	RC5	Simon	Speck	SIT	PROPOSED
Program Memory	17.62	10.28	2.82	1.33	15.29	1.05	1.95	0.63	0.94
Data Memory	17.58	7.03	10.55	6.69	8.79	4.59	3.03	0.54	0.83

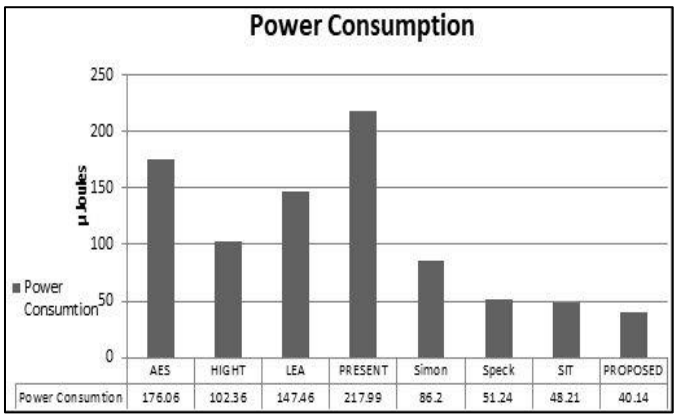
Fig. 22. The comparison of data and memory usage of ciphers.

If the CPU cycle is known, then the energy consumption of the algorithm can be measured. The equation [18] as follows:

$$E = I * VCC * \tau * N \quad (7)$$

Here, VCC is the supply voltage of the system and I is the average current in amperes in which is consumed of T seconds. τ is the clock period and N is the number of clock cycle. So clock period is $\tau = {}^{1}/_{f}\ sec/cycle$.

Atmel Atmega128 generally uses operating voltage in range of 2.7~5.5, current 40mA on average, and also operates at 16 MHz. The figure 23 shows comparison of power consumption among existing ciphers with the proposed cipher.



	AES	HIGHT	LEA	PRESENT	Simon	Speck	SIT	PROPOSED
Power Consumption	176.06	102.36	147.46	217.99	86.2	51.24	48.21	40.14

Fig. 23. Energy consumption comparison of ciphers.

V. Conclusion And Future Work

In the near future resource-constraint devices will be essential element of everybody's daily lives with the blessing of modern electronics and internet. Those devices will be communicating with each other incessantly, so security of the data must be considered. For this purpose an effective

lightweight cryptography algorithm proposed in this paper, with a **reduced 16.73% power consumption** than the existing cipher. The implementation shows promising performance making the algorithm a suitable candidate for resource-constraint devices. For future an indomitable challenge is taken to reduce computation cycle for sophisticated resource-constrained devices In-shah-Allah.

REFERENCES

[1] K. A. McKay, L. Bassham, M. S. Turan, N. Mouha, "Report on Lightweight Cryptography", National Institute of Standards and Technology, USA, March 2017.

[2] B. Schneier, Applied Cryptography: protocols, algorithms, and source code in C, john wiley & sons, 2007.

[3] L. Khelladi, Y. Challal, A. Bouabdallah, N. Badache, "On Security Issues in Embedded Systems: Challenges and Solutions", International Journal of Information Security, Inderscience, 2008, 2 (2), pp.140-174.

[4] S. A. Kumar, T. Vealey, and H. Srivastava, "Security in Internet of Things: Challenges, solutions and future directions", in 2016 49th Hawaii International Conference on System Sciences (HICSS), IEEE, 2016, pp.5772-5781.

[5] H.J. Ban, J. Choi, and N. Kang, "Fine-grained support of security services for resource constrained internet of things", International Journal of Distributed Sensor Networks, vol. 2016, 2016.

[6] P. Wang, Professor S. Chaudhry, S. Li, T. Tryfonas and H. Li, "The internet of things: a security point of view", Internet Research, vol. 26, no. 2, pp. 337-359, 2016.

[7] S. Wang, Z. Zhang, Z. Ye, X. Wang, X. Lin, and S. Chen, "Application of environmental internet of things on water quality management of urban scenic river", International Journal of Sustainable Development & World Ecology, vol. 20, no3, pp. 216-222, 2013.

[8] William Stallings. "Cryptography and Network Security Principles and Practices", Fourth Edition, Publisher: Prentice Hall, November 16, 2005

[9] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, and T. Phillips, "Guide-lines for securing radio frequency identification (RFID) systems", NIST Special publication, vol. 80, pp. 1-154, 2007.

[10] Shreyank N Gowda," Innovative Enhancement Of The Caesar Cipher Algorithm For Cryptography" 978-1-5090-3480-2/16/$31.00 ©2016 IEEE

[11] Vikash Kumar Jha," Cryptanalysis of Lightweight Block Ciphers" Aalto University School of Science Degree Programme of Computer Science and Engineering, Master's Thesis, November 18, 2011

[12] J. Gitanjali, Dr. N. Jeyanthi, C. Ranichandra, M. Pounambal, "ASCII Based Cryptography Using Unique id, Matrix Multiplication and Palindrome Number" School Of Information Technology and Engineering, VIT University, India

[13] S. Dutta, T. Das, S. Jash, D. Patra, Dr. P. Paul, "A Cryptography Algorithm Using the Operations of Genetic Algorithm & Pseudo Random Sequence Generating Functions", International Journal of Advances in Computer Science and Technology, ISSN 2320 – 2602, Volume 3, No.5, May 2014

[14] M. Usman, I. Ahmed, M. I. Aslam, S. K. and U. A. Shah, "SIT: A Lightweight Encryption Algorithm for Secure Internet of Things", Iqra University, Defence View and Department of Electronic Engineering, International Journal of Advanced Computer Science and Applications, Vol. 8, No. 1, 2017.

[15] D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. L. Corre, L. Perrin, "FELICS – Fair Evaluation of Lightweight Cryptographic Systems", University of Luxembourg, July 2015.

[16] Webster and S.E. Tavares, "On the design of s-boxes", in Conference on the Theory and Application of Cryptographic Techniques. Springer, 1985, pp. 523-534.

[17] E. Shannon, "Communication theory of secrecy systems", Bell system technical journal, vol. 28, no. 4, pp. 656-715, 1949.

[18] MOJTABA ALIZADEH, MAZLEENA SALLEH, MAZDAK ZAMANI, JAFAR SHAYAN, SASAN KARAMIZADEH. "Security and Performance Evaluation of Lightweight Cryptographic Algorithms in RFID", Faculty of Computer and Information Systems, Advanced Informatics School Universiti Teknologi Malaysia.