Energy Efficient Encryption Algorithm for Wireless Sensor Network

A. Babu Karuppiah[1], Dr. S. Rajaram[2]
[1] *Velammal College of Engineering and Technology, Madurai, India*
[2]*Thiagarajar College of Engineering, Madurai, India*

Abstract

Sensor devices have critical resource constraints such as processing speed, memory size and energy supply. Especially, energy consumption affects the network lifetime so that energy efficiency is an important requirement for Wireless Sensor Networks (WSNs). It means that it is a considerable matter to choose the energy- and a memory-efficient cryptographic algorithm suitable for WSNs. In this paper, an Energy Efficient Encryption Algorithm with 64-bit block length and 128-bit key length with basic operations like exclusive-OR (XOR) and shifting is implemented. It provides low-resource hard-ware implementation, which is suitable to sensor devices. It not only consists of simple operations but also has enough security as a good encryption algorithm. The reduction in the hardware resources also decreases the power consumption of the Wireless Sensor Network.

Keywords - Wireless Sensor Networks (WSNs), Encryption algorithm, Energy Efficiency.

1. Introduction

Wireless Sensor Networks consist of base station and a lot of battery-powered and low-cost devices, called sensor nodes. Distributed Wireless Sensor Network consist of several scattered nodes in a knowledge area. Those sensors have as its only power supplies a pair of batteries that must let them live up to five years without substitution. That's why it is necessary to develop some power aware algorithms that could save battery lifetime as much as possible. WSNs have currently been used for a variety of applications such as environment monitoring, health monitoring, military applications, etc. and also WSNs are expected to be used at anywhere in the near future. To support many kinds of applications based on sensor networks, the consideration of security aspects such as Data Confidentiality, Data Integrity, and Data Authenticity is essential [1]. Since the interest lies in implementing a security scheme for wireless sensor networks, here it is a must to choose the schemes that meet the limited resources of sensor nodes. The key issue of designing cryptographic algorithms is to deal with the trade-off among security, cost, and performance. A host of cryptographic primitives that particularly target resource-constrained smart devices have been published in the past few years and focus will be on lightweight symmetric ciphers in this paper. All the previous proposals can be roughly divided into the following three categories. The first category consists of highly optimized and compact hardware implementations for standardized block ciphers such as AES. However, the energy consumption of strong encryption is relatively high, whereas the proposals in the second category involve slight modifications of a classical block cipher like DES for lightweight applications. Finally, the third category features new low-cost designs, including lightweight block ciphers HIGHT. Thus Public key encryption algorithm is a fundamental and widely used technology around the world, since it has hardware limitations like memory usage and battery life, so it is not applied to the sensor network. Therefore, Symmetric key encryption algorithm with Low- Energy consumption is used in the sensor networks. In this paper, a new block cipher encryption algorithm is presented which has a 32-round iterative structure which is a variant of generalized Feistel network. The prominent feature of the algorithm is that it consists of simple operations such as XOR, addition mod 28, and left bitwise rotation. So, it is hardware-oriented rather than software-oriented. Moreover, the proposed algorithm is implemented using verilog language in Xilinx software and the experiment results shows that this algorithm is more energy efficient than the existing algorithms, the hardware resources consumption of the existing algorithm such as HIGHT [4] is more than the proposed encryption algorithm in the way as number of gate usage, total CPU time to Xst completion, total memory usage, number of slices used etc. So these result analyses provides us criterion of selecting Energy Efficient Encryption Algorithm suitable for WSNs.

This paper is organized as follows: In section II, issues in wireless sensor networks are described. In section III, Security requirements in WSN are explained. In section IV, the existing encryption algorithm schemes are presented. In section V, the proposed encryption algorithm scheme is analyzed how efficient it can be and whether it can meet the low power resource or not. In the final section, the experimental results are discussed followed by concluding remarks and references.

2. Issues in WSNs

A WSN is a network of cheap and simple processing devices (sensor nodes) that are equipped with environmental sensors for temperature, humidity, etc. and can communicate with each other using a wireless radio device. Sensor networks need to become autonomous and exhibit responsiveness without explicit user or administrator action. Security has become a significant issue in regards to the violation of individual information security [2, 3]. Encryption is supposed to make a secret document that is functionally different from the original document. As WSNs grow to be more popular and widely used, security becomes a very serious concern. Users do not want to reveal their data to unauthorized people as the disclosed information could be used for malicious purposes. This concern is even more relevant to wireless environments where anyone can overhear a message sent over the radio. Therefore, even a very useful and convenient system might not be appealing to the users if it is not secure. Security is in general considered to be expensive. Its cost is even more noticeable in WSNs due to the limited resources of the sensor nodes. Thus, in order to provide a sufficient level of security while properly utilizing the available resources, it is important to have a good understanding of both the cost and the features of the security algorithms used. Apart from the security issue, the primary challenges for sensor networks stem from two facts [2]. First, sensors are extremely resource constrained. Second, in many applications sensor nodes will be randomly deployed. The resource limitation of sensor networks poses great challenges for security. As sensor nodes are with very limited computing power, it is difficult to provide security in WSN using public-key cryptography.

3. Security Requirements in WSNs

Recall that the most important network services that should be implemented by any security mechanism are: data confidentiality, data Integrity, and data Authenticity [5]. Since interest lies in implementing a security scheme for wireless sensor networks, the schemes must be chosen that meet the limited resources of sensor nodes. In this section the reasons for choosing the schemes are discussed. The process by which public key and symmetric key cryptography schemes should be selected is based on the following criteria:

Energy : how much energy is required to execute the encryption/decryption functions.

Program memory : the memory required to store the encryption/decryption program

Temporary memory : the required RAM size or number of registers required temporarily when the encryption/decryption code is being executed.

Execution time : the time required to execute the encryption/decryption code.

4. Existing Algorithms for Encryption

A. AES Algorithm

The Advanced Encryption Standard (AES) algorithm, also known as Rijndael, is a block cipher adopted as an encryption standard by the US government. AES is a substitution– permutation network and is one of the most popular symmetric-key cryptography algorithms. AES is fast in both software and hardware and is relatively easy to implement. It operates on a 4 by 4 array of bytes and has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits with 10, 12, and 14 number of rounds [3]. For encryption, each round of AES (except the last round) consists of four stages. a) SubBytes - a non-linear substitution step where each byte is replaced with another according to a lookup table (known as S Box). b) ShiftRows - a transposition step where each row of the state is shifted cyclically a certain number of steps. c) MixColumns – a mixing operation which operates on the columns of the state, combining the four bytes in each column using a linear transformation. d) AddRoundKey - each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule. AES algorithm comprises of various rounds depending on the key size and block size. Out of all the rounds the Pre- round comprises only AddRoundKey whereas the final round omits the MixColumns stage.

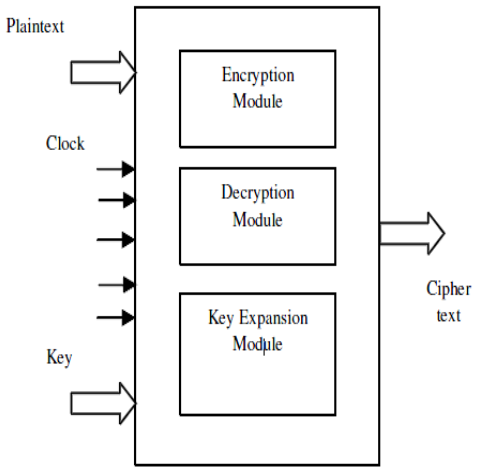


Figure. 1 Hardware I/O Specifications

Figure. 1 shows the encryption process of the AES algorithm. Here the plaintext is applied along

with the key and clock input to obtain the ciphertext by applying operations with encryption and decryption module. The round transformation modifies the 128-bit state. The initial state is the input plaintext and the final state is the output ciphertext where the state is organised as a 4 X 4 matrix of bytes.

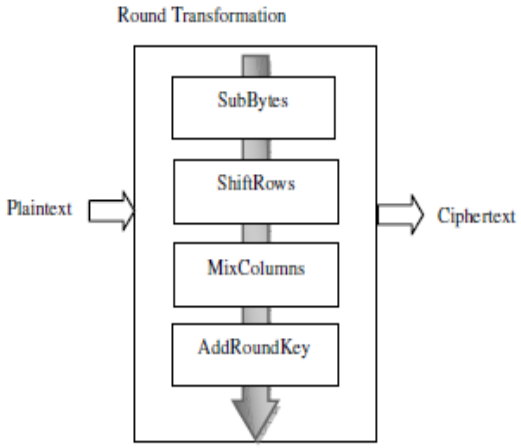


Figure. 2 AES Round Transformation

Figure. 2 shows the round transformation of AES, where it scrambles the bytes of the state either individually, rowwise, or columnwise by applying the functions SubBytes, ShiftRows, MixColumns, and AddRoundKey sequentially [7].

B. RC5 Algorithm

RC5 is a fast symmetric block cipher suitable for hardware or software implementations. RC5 has a variable block size (32, 64 or 128-bits), key size (0 to 2040 bits) and number of rounds (0 to 255) [6]. The original suggested choices of parameters were a block size of 64-bits, a 128-bit key and 12-rounds. It is fast symmetric block cipher suitable for hardware or software implementations. A novel feature of RC5 is the heavy use of data-dependent rotations. RC5 has a variable word size, a variable number of rounds, and a variable-length secret key. The encryption and decryption algorithms are exceptionally simple. RC5 is not intended to be secure for all possible parameter values. On the other hand, choosing the maximum parameter value will be overkill for most applications. RC5 is hard to use in open environments and one-shot communications. 12-round RC5 (with 64-bit blocks) is susceptible to a differential attack using 244 chosen plaintexts. 18-20 rounds are suggested as sufficient protection. A distinguishing feature of RC5 is its heavy use of data dependent rotation. The amount of rotation performed is dependent on the input data and it's not pre-determined. Word size, number of rounds and key size of RC5 can be varied. The different combinations of values for these parameters are used to fully understand their influence on the energy consumption caused by the encryption algorithm:

1. The usual word size for encryption is 32 bits (4 bytes) to study the impact of the word size on the time it takes to perform key setup, encryption, and decryption.
2. The number of rounds (4,8,12,16,18) has a proportional effect on the security of RC5 [10].
3. RC5 also uses different key sizes as AES.

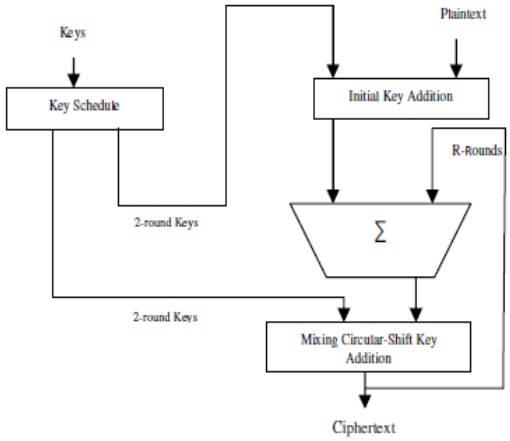


Figure. 3 Block Diagram of RC5

Figure. 3 shows the Block Diagram of RC5 that illustrates the functioning with initial key addition, Mixing Circular Shift Key Addition blocks to obtain ciphertext by applying the plaintext and keys.

C. Skip Jack

Skipjack uses an 80-bit key to encrypt or decrypt 64-bit data blocks. It is an unbalanced Feistel network with 32 rounds [6]. It has been found that the number of rounds is exponentially proportional to the amount of time required to find a key using a brute-force attack. So, as the number of rounds increases, the security of the algorithm increases exponentially. Skipjack uses F-BOX which can be stored in either RAM or program memory.

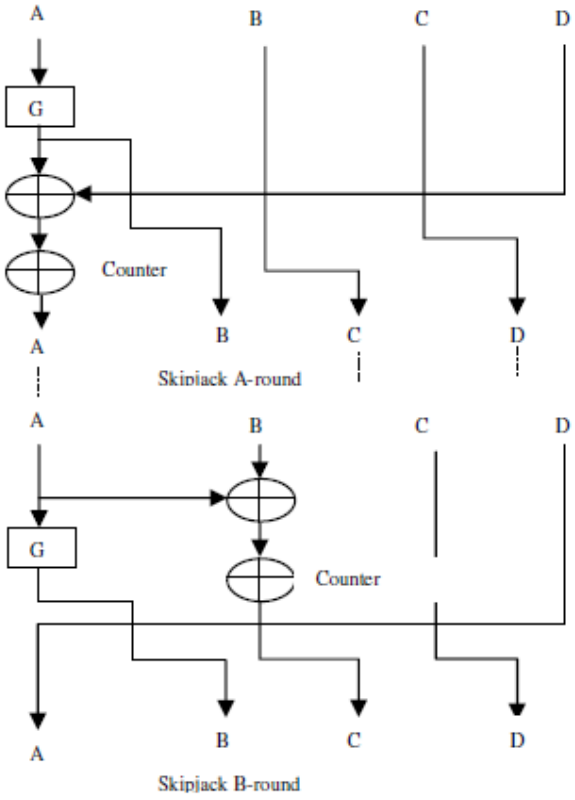


Figure. 4 Skip Jack Round Function

From Figure 4, the output of the Skipjack Round function can be explained with the following expressions,

A (a,b,c,d) = (d + Gk(a) + counter, GK (a), b,c);
B (a,b,c,d) = (d;Gk (a); a + b + counter; c):
A^{-1}(a,b,c,d) = (G−1k (b); c; d; a + b + counter);
B^{-1} (a,b,c,d) = (G−1k (b); c + G−1k (b) + counter; d; a):

Skipjack has been extensively crypt analyzed, and has no weaknesses. There are no known shortcut attacks that can break Skipjack. However, the small key size makes this algorithm inferior to the newer candidate algorithms for the Advanced Encryption Standard (AES) competition being held by NIST.

D. HIGHT

HIGHT has 64-bit block length and 128-bit key length, which is suitable for low-cost, low power and ultra-light implementation. 32-round iterative structure which is a variant of generalized Feistel network [6]. The hardware implementation of HIGHT requires 3048 gates on 0.25 µm technology [4]. It has been analyzed for the security against various attacks. The strength of the HIGHT algorithm is evaluated with respect to differential attack, linear attack, truncated differential cryptanalysis, impossible differential cryptanalysis, saturation attack, boomerang attack, interpolation and higher order differential attack.

5. Proposed Method

The existing HIGHT Algorithm utilizes more hardware resource by using large number of X-OR gates for all computational rounds. By reducing the number of gates, resource utilization is also reduced which in turn reduces power consumption which is achieved by the concept of reusability. Here the concept of reusability in verilog coding is utilized that uses a single output variable for all round iterations, which results in only one X-OR gate that is used for all round operations instead of as many X-ORs as rounds. The 64-bit plain-text and ciphertext are considered as concatenations of 8 bytes and denoted by P = P7||……P1||P0 and C = C7||.….C1||C0, respectively. The 64-bit intermediate values are analogously represented, Xi = $X_{i,7}$||…….$X_{i,1}$||$X_{i,0}$ for i = 0,…..,32. The 128-bit master key is considered as a concatenation of 16 bytes and denoted by MK = MK15||……||MK0. The following are notations for mathematical operations:
WK – Whitening Key; SK – SubKey

⊞ : Addition mod 2^8
⊟ : Subtraction mod 2^8
⊕ : XOR (Exclusive OR)
$A^{<<s}$: s-bit left rotation of a 8-bit value A

The detailed process of Encryption of HIGHT Algorithm is depicted in Figure. 5.

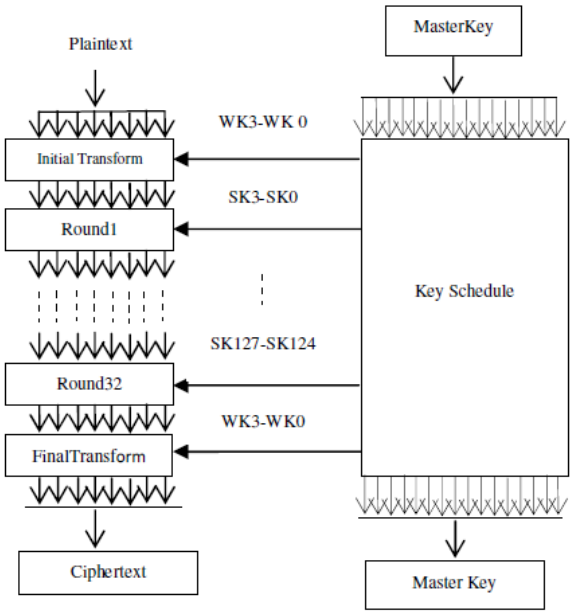


Figure. 5 Encryption Process of HIGHT Algorithm

HIGHT Encryption algorithm consists of key schedule, initial transformation, round function, and final transformation.

```
HightEncryption(P,MK)
{
KeySchedule(MK,WK,SK);
HightEncryption(P,WK,SK)
{
InitialTransfomation(P,X0,WK3,WK2,WK1,WK0);
For i = 0 to 31 {
RoundFunction(Xi,Xi+1, SK4i+3, SK4i+2, SK4i+1, SK4i);
}
FinalTransfomation(X32,C,WK7,WK6,WK5,WK4);
}
}
```

A. Whitening Key Generation

It uses 8 whitening key bytes WK0,…..,WK7 for the initial and final transformations. The algorithm WhiteningKeyGeneration generates them as follows.

```
WhiteningKeyGeneration {
For i = 0 to 7 {
If (0< i < 3) then WKi ← MKi+12;
Else, WKi ← MKi-4;
}
}
```

It is observed that Whitening Keys (WK0,….,WK7) are generated from the 16 bytes input Master Key. This serves as a Key for the Initial Transformation Block where the operation is done with plaintext and whitening keys and proceeded further.

B. Initial Transformation

Initial Transformation transforms a plaintext P into the input of the first Round Function, $X_0 = X_{0,7}\|X_{0,6}\|\ldots.\|X_{0,0}$ by using the four whitening-key bytes, WK_0, WK_1,WK_2, and WK_3.

C. Subkey Generation

128 subkeys are used for 1 computation of HightEncryption, 4 subkeys per round. The algorithm SubkeyGeneration uses the subalgorithm ConstantGeneration to generate 128 7-bit constants δ_0 ,…….,δ_{127}, and then generates the subkeys SK_0,……,SK_{127} with the constants. δ_0 is fixed as 1011010_2. This is also the initial state (s_6 ,……, s_0) of 7-bit LFSR and so $\delta_0=\delta_{127}$.

```
SubkeyGeneration(MK,SK) {
      Run Constant Generation
     For i = 0 to 7 {
   For j= 0 to 7 {
      SK16.i+j ← MKj-imod8 ⊞δ16.i+j ;
   }
```

```
   For j = 0 to 7 {
      SK16.i+j ← MK(j-imod8) + 8 ⊞δ16.i+j+8 ;
   }
    }
}
```

D. Round Function

RoundFunction uses two auxiliary functions F0 and F1:

$$F_0(x) = x^{<<1} \oplus x^{<<2} \oplus x^{<<7};$$

$$F_1(x) = x^{<<3} \oplus x^{<<4} \oplus x^{<<6};$$

For i=0,……..,31, RoundFunction transforms $X_i=X_{i,7}\|\ldots,\|X_{i,0}$ into $X_{i+1} =X_{i+1,7}\|\ldots,\|X_{i+1,0}$ as follows.

```
For(i=0;i<=32;i=i+1) {
RoundFunction(Xi,0, Xi+1,SK4i+2,SK4i+1,SK4i) {
   X1←X0; X3 ←X2; X5 ←X4; X7 ←X6;
   X0=X7⊕ (F0(X6) ⊞SK4i+3);
   X2=X1⊞ (F1(X0) ⊕SK4i+2);
   X4=X3⊕ (F0(X2) ⊞SK4i+1);
   X6=X5⊞ (F1(X4) ⊕SK4i);
   }
}
```

The above algorithm is used for generation of all 32 round function block. The output of the preceding round is given as the input to the successive round.

E. Final Transformation

```
InitialTransformation(P,X0,WK3,WK2,WK1,WK0){
X0,0 ← P0⊞ WK0; X0,1← P1 ;
X0,2← P2⊕ WK1;
X0,3 ←P3 ;
X0,4 ← P4⊞ WK2; X0,5 ← P5 ;
X0,6← P6⊕ WK3;
X0,7 ← P7 ;                                    }
```

FinalTransformation untwists the swap of the last round function and transforms $X_{32} = X_{32,7}\|X_{32,6}\|,\ldots,\|X_{32,0}$ into the ciphertext C by using the four whitening-key bytes WK_4, WK_5, WK_6, and WK_7. This step is similar to initial Transformation. It is observed that the X-OR and modular arithmetic operations are performed to generate 7-byte ciphertext.

```
FinalTransformation(C,X32,WK7,WK6,WK5,WK4) {
C0 ← X32,1⊞ WK4; C1 ← X32,2 ; C2←X32,3⊕
```

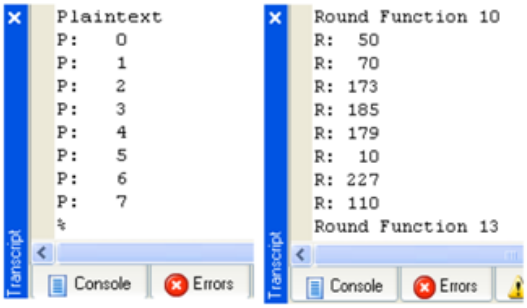
WK_5;
$C_3 \leftarrow X_{32,4}$;
$C_4 \leftarrow X_{32,5} \boxplus WK_4$; $C_5 \leftarrow X_{32,6}$; $C_6 \leftarrow X_{32,7} \oplus$
WK_5;
$C_7 \leftarrow X_{32,0}$;
}

F. Decryption

The decryption operation is identical in operation to encryption apart from the following two modifications. (1) All addition modular 2^8 operations are replaced by subtraction modular 2^8 operations except for the operations connecting SK_i and outputs of F_0.
(2) The order in which the keys WK_i and SK_i are applied is reversed.

6. Experimental Results and Comparison

The structure of HIGHT is generalized Feistellike. This kind of structure reduces restriction of designing inner auxiliary functions. Compared to AES like structure, the round function is light. Since encryption process is simply converted into decryption process, implementation of the circuit supporting both encryption and decryption processes does not require much more cost than the encryption-only circuit. Every operation in HIGHT is 8- bit-processor-oriented. CPUs embedded into the sensors in USN (Ubiquitous Sensor Network) are based on 8-bit processor. So, HIGHT has efficient performance in such environment.



(a) (b)

Figure.7 (a) Generation of Plain text (b) A sample Round function block

Figure 7 (a) shows the generation of plaint text and (b) shows the generation of sample round function block.

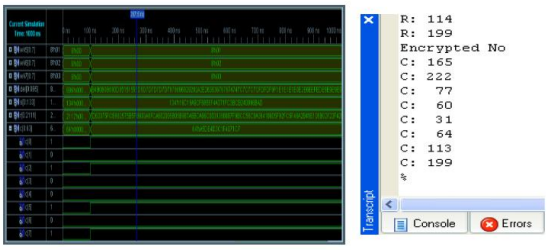


Figure.8 Simulation result for Encryption

Figure 8 shows the simulation result of the encryption algorithm for 64-bit plaintext, 128 Subkeys, 8 Whitening Keys using 32 rounds. The target device chosen was XC3S250E of Spartan3E family. It shows the 64-bit encrypted result using Energy Efficient Encryption Algorithm.

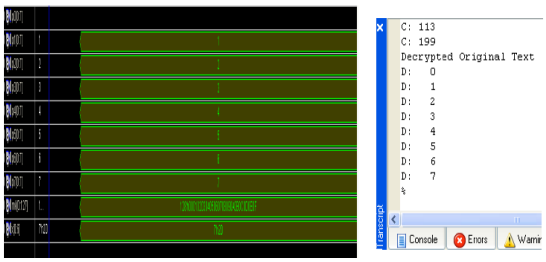


Figure.9 Simulation result for Decryption

Figure 9 shows the simulation result of the decryption process where the original 64-bit plaintext is retained, using the same Whitening Keys and SubKeys as in encryption.

Table1: Comparison of Results

S. No.	Parameters Taken	Existing architecture	Proposed architecture
1.	Number of Slices	62%	51%
2.	Number of 4 input LUTs	58%	48%
3.	Number of IOs:	4493	2438

Table 1 shows the comparison of results that are obtained from the synthesis report. It can be inferred that the proposed algorithm has an advantage over the existing algorithm by consuming 11% less number of slices, 10% less number of 4 input LUTs and a greater reduction in the number of inputs and outputs.

7. Conclusion

For any Wireless Sensor Network the battery life time and the energy of the network are limited. In this paper, it is shown that the proposed Energy Efficient Encryption Algorithm which includes the

basic operations of XOR and shift is suitable to overcome the limitations of wireless sensor nodes. The result indicates that the algorithm achieves reduction in hardware resources providing energy efficiency thereby increasing the life span of wireless sensor nodes in the network.

8. References

[1] Sang-Eon Lee, Sang-Ho Shin, Geum-Dal Park and Kee-Young Yoo, "Wireless Sensor Network Protocols for Secure and Energy-Efficient Data Transmission" Proceedings of the CISIM'08 on Computer Information Systems and Industrial Management Applications, 26-28 June 2008.

[2] Zoran S. Bojkovic, Bojan M. Bakmaz, and Miodrag R. Bakmaz "Security Issues in Wireless Sensor Networks" in International Journal of Communications, Issue 1, Volume 2, pp.1

[3] Jongdeog Lee, Krasimira Kapitanova and Sang H. Son, "The price of security in wireless sensor networks", Computer Networks 54 (2010) pp.2967–2978.

[4] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim and Sangjin Lee "HIGHT: A New Block Cipher Suitable for Low-Resource Device", Proceedings of the Cryptographic Hardware and Embedded Systems (CHES) Volume 4249, in 2006, pp.46-59

[5] Mohammad AL-Rousan, A. Rjoub and Ahmad Baset, "A Low-Energy Security Algorithm for Exchanging Information in Wireless Sensor Networks" Published in Journal of Information Assurance and Security4 (2009), pp.48-59.

[6] Woo Kwon Koo, Hwaseong Lee, Yong Ho Kim and Dong Hoon Lee, "Implementation and Analysis of New Lightweight Cryptographic Algorithm Suitable for Wireless Sensor Networks", Proceeding of International Conference on Information Security and Assurance, 2008.

[7] Sounak Samanta, "FPGA Implementation of AES Encryption and Decryption", in an International Conference on "Control, Automation, Communication and Energy Conservation, June 2009.

[8] Jason Van Dyken and José G. Delgado-Frias, "FPGA Schemes for Minimizing the Power-Throughput Trade-off in Executing the Advanced Encryption Standard Algorithm", Published in Journal of Systems Architecture 56 (2010), pp.116–123.

[9] Gil-Ho Kim, Jong-Nam Kim and Gyeong-Yeon Cho, "An improved RC6 algorithm with the same structure of encryption and decryption", Proceeding of ICACT on 11th International Conference of Advanced Communication Technology, 2009, Volume 2, pp.1211 – 1215.

[10] B.S. Kaliski Jr., Y.L. Yin, On the security of the RC5 encryption algorithm, RSA Laboratories, September 2006, TR-602, Version 1.0.

[11] ASIC Official Website : http://www.asic-world.com