Ilkogretim Online - Elementary Education Online, 2020; Vol 19 (Issue 3): pp. 4346-4356
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2020.03.735586

A Structured Encrypted Authentication For Wireless Sensor Networks

[1]Mrs. Aswini V , [2]Mr. Aashrith ,[3] Mr. KrishnaTeja[4] , Mr.Dharanidharan

[1]Assistant Professor, Sri Sai Ram Institute of Technology, Sai Leo Nagar, West Tambaram, Chennai

[2]Student, Sri Sai Ram Institute of Technology, Sai Leo Nagar, West Tambaram, Chennai

[3]Student, Sri Sai Ram Institute of Technology, Sai Leo Nagar, West Tambaram, Chennai

Abstract: The paper presents a Network security is very important in mobile devices in which different types of cryptographic algorithms are used to prevent malicious attack on the transmitted data. This work provides an encryption mechanism based on Rabbit stream cipher for providing confidentiality in Wireless Sensor Networks. It is a software oriented stream cipher with very strong security properties and support for 128-bits keys. It can be widely used in the applications of secure communication nodes that have limited processing and storage capabilities. Based on the implementation result it can be seen that the rabbit stream cipher algorithm is better than block cipher algorithm with regards to CPU time required for encryption. We propose the idea of using rabbit cipher to make this happen.

Index terms: Encryption, Cipher Security, Symmetric Key, Authentication, Confidentiality, Integrity, Initialization Vector, Rabbit-MAC, and Wireless Sensor Network.

I. INTRODUCTION

A software - oriented synchronous stream cipher called Rabbit[8] which includes very strong security properties and supports a 128-bit key. Rabbit is part of the portfolio of stream ciphers addressing the need of strong and efficient computation with fast ciphers. The Stream cipher is designed with both security and efficiency to satisfy the need for lightweight algorithms, dedicated to hardware environments where the available resources are heavily restricted.

Wireless Sensor Networks (WSNs) consist of small resource constrained devices termed as nodes, which combine an 8-bit processor with memory, sensors, radio unit and power supply, called sensor nodes that collect data from their environments and then collaborate to transmit the data on to a sink node (base station) [1].

WSN requires thousands of battery-powered devices which communicate wirelessly. To lower the cost, these devices are constrained in terms of memory capacity, computing power and energy supply.

In many applications security plays a vital role for performance and low energy consumption.

Security is used on premise security and surveillance, building monitoring, burglar alarms, disaster recovery and in critical systems such as airports, hospitals. Symmetric-key algorithms are less computation and intensive than symmetric key algorithms.

The aspect of this paper with different applications as shown in fig. 1, is organized as follows: In Section II, it presents the security mechanisms that are guaranteed for the proposed scheme.In Section III, it explains the Rabbit Stream Cipher Algorithm Specifications and the appropriateness of utilizing Rabbit for WSN security. In Section IV, it presents the detail of working and analysis of its cost effectiveness for security in WSNs. Then, section V shows the implementation work and finally, Section VI draws the conclusions of the proposed scheme.

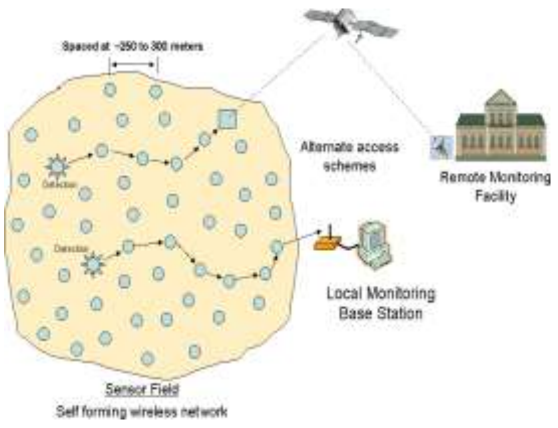

Spaced at ~250 to 300 meters
Detection
Detection
Alternate access schemes
Remote Monitoring Facility
Local Monitoring Base Station
Sensor Field
Self forming wireless network

Figure 1. WSNs used in Military Applications

II. SECURITY MECHANISMS

The Rabbit-MAC scheme guarantees the authenticity, integrity, and confidentiality of messages between neighbouring nodes, while permitting in network processing. The Security Mechanisms are discussed below:

2.1 Message Confidentiality

Confidentiality ensures to keep the information such as data, resources etc., as a secret from unauthorized parties and achieve using with encryption. A sensor network should not leak sensor readings to adversaries. In many applications (e.g. key distribution) nodes communicate highly sensitive data. Preferably, an encryption scheme should not only prevent message recovery, but also prevent adversaries from learning even partial information about the messages that have been encrypted using some sort of encryption scheme.

2.2 Message Authentication

Message authentication is important for many applications in sensor networks. In critical WSN applications, the receiver needs to verify that the data originated from an authorized sensor node. Message authentication means that the link layer security protocol should prevent unauthorized parties from participating in the network and that legitimate nodes within the network should reject unauthorized packets as soon as they enter the network, to save unnecessary energy and bandwidth wastage.

2.3 Message Integrity

Data integrity ensures the receiver that, the received data is not altered in transit by an adversary and that the message is exactly what was sent by the sender of the message. WSNs use wireless broadcasting as communication method. Thus it is more vulnerable to eavesdropping and message alteration, since the message is broadcasted. Measures for protecting integrity are needed to detect message alteration and to reject injected message that have been placed into the originally transmitted message.

III. THE DESIGN OF RABBIT CIPHER SCHEME

Rabbit cipher scheme was designed to be faster cipher and to justify a key size of 128 bits for encryption up to 2^{64} bytes of plaintext. This means that for an attacker who does not know the key, it should not be possible to distinguish up to 2^{64} bytes of cipher output from the output of a truly random generator, using less steps than would be required for an exhaustive key search over 2^{128} keys [10]. Rabbit takes a 128- bit secret key and a 64-bit IV (if desired) as input and generates for each iteration an output block of 128 pseudo-random bits from a combination of the internal state bits. Encryption/decryption is done by XOR'ing the pseudo-random data with the plaintext/ciphertext. The size of the internal state is 513 bits divided between eight 32-bit state variables, eight 32-bit counters and one counter carry bit. The eight state variables are updated by eight coupled nonlinear functions.

3.1 Notation

The notation used in algorithm are: $\oplus$ denotes logical XOR, & denotes logical AND, << and >>denote left and right logical bit-wise shift, <<< and >>> denote left and right bit- wise rotation, and || denotes concatenation of two bit sequences. It means bit number g through h of variable A. When numbering bits of variables, the least significant bit is denoted by 0. Hexadecimal numbers are prefixed by ”0x”.

3.2 The Rabbit Cipher Algorithm

The internal state of the stream cipher consists of 513 bits. 512 bits are divided between eight 32-bit state variables xj,i and eight 32-bit counter variables cj,i, where xj,i is the state variable of subsystem j at iteration i, and cj,i denotes the corresponding counter variable. There is one counter carry bit, Ø7,i, which needs to be stored between iterations. This counter carry bit is initialized to zero. The eight state variables and the eight counters are derived from the key at initialization.

3.2.1 Key setup Scheme

The algorithm is initialized by expanding the 128-bit key into both the eight state variables and the eight counters such that there is a one-to-one correspondence between the key and the initial state variables, $x_{j,0}$, and the initial counters,

$c_{j,0}$. The key, $K[127..0]$, is divided into eight sub keys: $k_0 = K[15..0]$,
$k_1 = K[31..16], \ldots, k_7 = K[127..112]$.
The state and counter variables are initialized from the sub keys as follows:

$x_{j,0} = k_{(j+1 \bmod 8)} \,||\, k_j$ for j even $k_{(j+5 \bmod 8)} \,||\, k_{(j+4 \bmod 8)}$ for j odd

and

$c_{j,0} = k_{(j+4 \bmod 8)} \,||\, k_{(j+5 \bmod 8)}$ for j even $k_j \,||\, k_{(j+1 \bmod 8)}$ for j odd

The system is iterated four times, according to the next-state function defined in section 3.3.3, to diminish correlations between bits in the key and bits in the internal state variables. Finally, the counter variables are re-initialized according to:

$c_{j,4} = c_{j,4} \oplus x_{(j+4 \bmod 8), 4}$

for all j, to prevent recovery of the key by inversion of the counter system.

3.2.3 IV Setup Scheme

Let the internal state after the key setup scheme be denoted the master state, and let a copy of this master state be modified according to the IV scheme. The IV setup scheme works by modifying the counter state as function of the IV. This is done by XORing the 64-bit IV on all the 256 bits of the counter state. The 64 bits of the IV are denoted IV [63..0].
The counters are modified as:

$c_{0,4} = c_{0,4} \oplus IV^{[31..0]}$ $c_{1,4} = c_{1,4} \oplus (IV^{[63..48]} || IV^{[31..16]})$

$c_{2,4} = c_{2,4} \oplus IV^{[63..32]}$ $c_{3,4} = c_{3,4} \oplus (IV^{[47..32]} || IV^{[15..0]})$

$c_{4,4} = c_{4,4} \oplus IV^{[31..0]}$ $c_{5,4} = c_{5,4} \oplus (IV^{[63..48]} || IV^{[31..16]})$

$c_{6,4} = c_{6,4} \oplus IV^{[63..32]}$ $c_{7,4} = c_{7,4} \oplus (IV^{[47..32]} || IV^{[15..0]})$.

The system is then iterated four times to make all state bits non- linearly dependent on all IV bits. The modification of the counter by the IV guarantees that all 264 different IVs will lead to unique key streams.

3.2.4 Next-State Function

The core of the Rabbit algorithm is the iteration of the system defined by the following equations:

$x_{0,i+1} = g_{0,i} + (g_{7,i} <<< 16) + (g_{6,i} <<< 16)$ $x_{1,i+1} = g_{1,i} + (g_{0,i} <<< 8) + g_{7,i}$

$x_{2,i+1} = g_{2,i} + (g_{1,i} <<< 16) + (g_{0,i} <<< 16)$ $x_{3,i+1} = g_{3,i} + (g_{2,i} <<< 8) + g_{1,i}$

$x_{4,i+1} = g_{4,i} + (g_{3,i} <<< 16) + (g_{2,i} <<<16)$ $x_{5,i+1} = g_{5,i} + (g_{4,i} <<< 8) + g_{3,i}$

$x_{6,i+1} = g_{6,i} + (g_{5,i} <<< 16) + (g_{4,i} <<<16)$ $x_{7,i+1} = g_{7,i} + (g_{6,i} <<< 8) + g_{5,i}$

$g_{j,i} =((x_{j,i} + c_{j,i+1})^2 \oplus ((x_{j,i} + c_{j,i+1})^2 >> 32)) \bmod 2^{32}$ where all additions are modulo 2^{32}.

3.2.5 Counter System

The dynamics of the counters is defined as follows: $c_{0,i+1} = c_{0,i} + a_0 + \emptyset_{7,i} \bmod 2^{32}$

$c_{1,i+1} = c_{1,i} + a_1 + \emptyset_{0,i+1} \bmod 2^{32}$ $c_{2,i+1} = c_{2,i} + a_2 + \emptyset_{1,i+1} \bmod 2^{32}$ $c_{3,i+1} = c_{3,i} + a_3 + \emptyset_{2,i+1} \bmod 2^{32}$ $c_{4,i+1} = c_{4,i} + a_4 + \emptyset_{3,i+1} \bmod 2^{32}$ $c_{5,i+1} = c_{5,i} + a_5 + \emptyset_{4,i+1} \bmod 2^{32}$

$c_{6,i+1} = c_{6,i} + a_6 + \emptyset_{5,i+1} \bmod 2^{32}$ $c_{7,i+1} = c_{7,i} + a_7 + \emptyset_{6,i+1} \bmod 2^{32}$

where the counter carry bit, $\emptyset_{j,i+1}$, is given by

1 if $c_{0,i} + a_0 + \emptyset_{7,i} \geq 2^{32} \wedge j = 0$

$\emptyset_{j,i+1}$ = 1 if $c_{j,i} + a_j + \emptyset_{j-1,i+1} \geq 2^{32} \wedge j > 0$

1 otherwise. 2

Furthermore, the a_j constants are defined as:

a_0 = 0x4D34D34D a_1 = 0xD34D34D3 a_2 = 0x34D34D34 a_3 = 0x4D34D34D a_4 = 0xD34D34D3 a_5 = 0x34D34D34 a_6 = 0x4D34D34D a_7 = 0xD34D34D3.

3.2.5 Extraction Scheme

After each iteration the output is extracted as follows:

```
 f[15..0]      [15..0] [31..16]
i     0, i     5, i
 f[31..16]     [31..16]          [15..0]
i     0, i     3, i
 f[47..32]     [15..0] [31..16]
i     2, i     7, i
 f[63..48]     [31..16]          [15..0]
i     2, i     5, i
 f[79..64]     [15..0] [31..16]
i     4, i     1, i
 f[95..80]     [31..16]          [15..0]
i     4, i     7, i
 f[111..96]    [15..0] [31..16]
i     6, i     3, i
 f[127..112]   [31..16]          [15..0]
i     6, i     1, i
```

where s_i is the 128-bit key stream block at iteration i.

3.2.6 Encryption / Decryption Scheme

The extracted bits are XOR'ed with the plaintext/ciphertext to encrypt/decrypt.

$c_i = p_i \oplus s_i$ $p_i = c_i \oplus s_i$

where c_i and p_i denote the ith 128-bit ciphertext and plaintext blocks, respectively.

3.2.7 Pseudo – random Number Generation Scheme

The extracted bits directly constitute the desired pseudo- random data.

$r_i = s_i$,

where r_i denotes the ith 128-bit pseudo-random number block.

IV RABBIT – MAC

Rabbit-MAC supports authenticated encryption, which provides message authentication and message integrity. Rabbit- MAC encrypts the data payload and authenticates the packet with a MAC. The MAC is computed over the encrypted data and the packet header. Due to the restrictions imposed on WSNs, our major objective in designing a new security model is to minimize cost-effect of the following while maintaining required levels of security [5]:

1. Communication overhead, in case of communicating with the encrypted packet.
2. Computation overhead, in securing the network, in order to save sensor's lifetime.
3. Utilized key space.

4.1 Rabbit -MAC Design

The Rabbit-MAC scheme was designed to conform to the security requirements of message authentication and message integrity. Our Rabbit-MAC employs Rabbit Next state function for calculating the MAC. This feature significantly reduces the excessive program space required by a separate MAC algorithm. It makes use of good diffusion and non-linearity properties of Rabbit next-state function. To send an authenticated packet, a sender simply computes a MAC on the packet with the agreed key MKey and encrypted message, as in fig 2:

Figure 2. Rabbit-MAC Process Diagram

When a node gets a packet, it can verify that the packet was sent by the corresponding node and no information has been altered in transit. This can be done by the receiving node computing the MAC with the agreed key Mkey and IV from the packet header and finally comparing it with the MAC received in the packet. Rabbit-MAC is an Encrypt-then-MAC stream cipher mode. Here the MAC algorithm reuses the next state function to produce MAC of encrypted message. Encrypt-then-MAC approach firstly provides message confidentiality and then message authentication.

4.2 IV Format

The IV is message-unique for the proposed Rabbit- MAC so that it will not give additional advantage to an attacker. The IV is taken from the packet header of the radio packet format and sent in clear to the decrypting party to save unnecessary wastage of resources in computing and transmitting the encrypted packet header.

4.3 Analysis

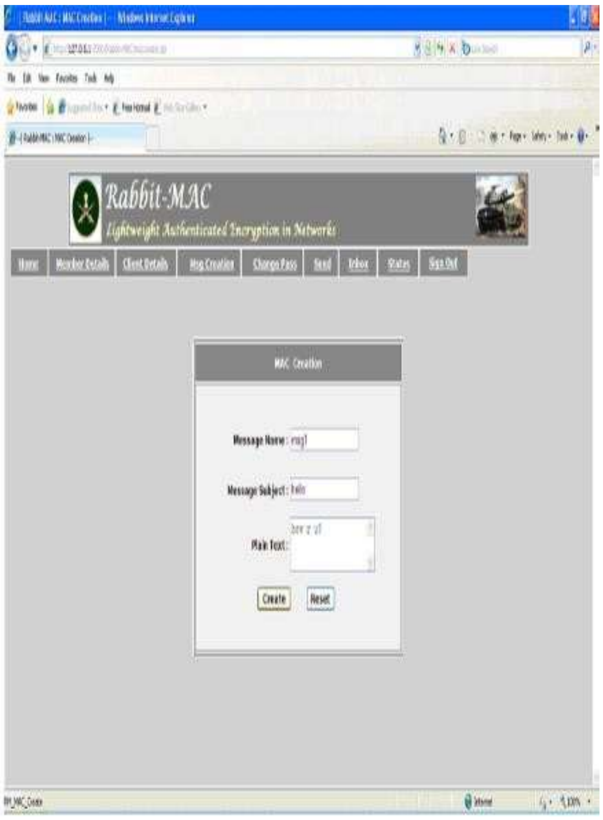
By employing the security architecture we can able to prevent most of the depicted threats while considering the constraints that the sensor network demands. It provides confidentiality as well as integrity for the communicated information and ensures authenticity of the sensor nodes. Confidentiality is achieved by encryption the messages. This prevents any illegitimate disclosure of information;

furthermore, Rabbit- MAC ensures the integrity and authenticity of the messages,. Within the constraints of WSNs.

V EXPERIMENTAL RESULT

The following screen shots shows how the data is moved from the sender to receiver.,

MAC Creation of Plain Text

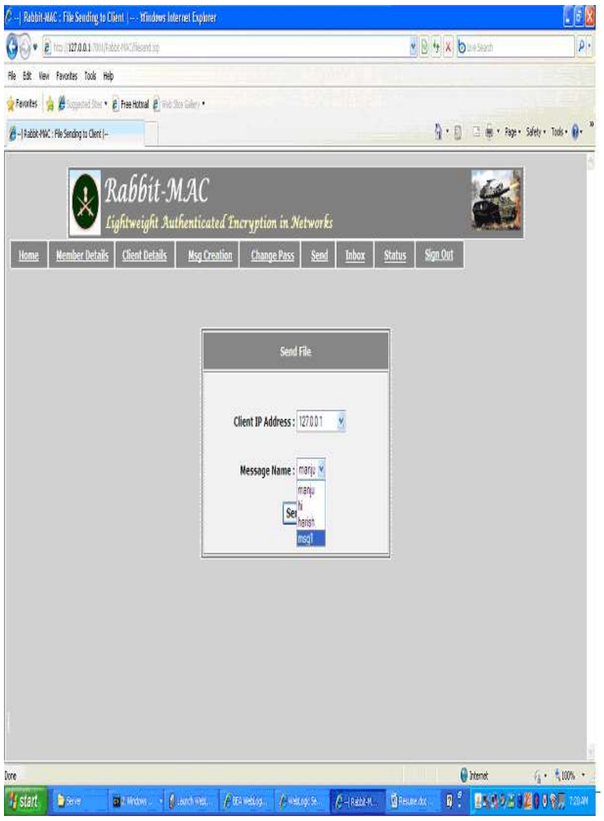


Message Name : msg1

Message Subject : hello

Plain Text : how r u?

Sender Sending Message


Rabbit-MAC : File Sending to Client

Rabbit-MAC
Lightweight Authenticated Encryption in Networks

Home | Member Details | Client Details | Msg Creation | Change Pass | Send | Inbox | Status | Sign Out

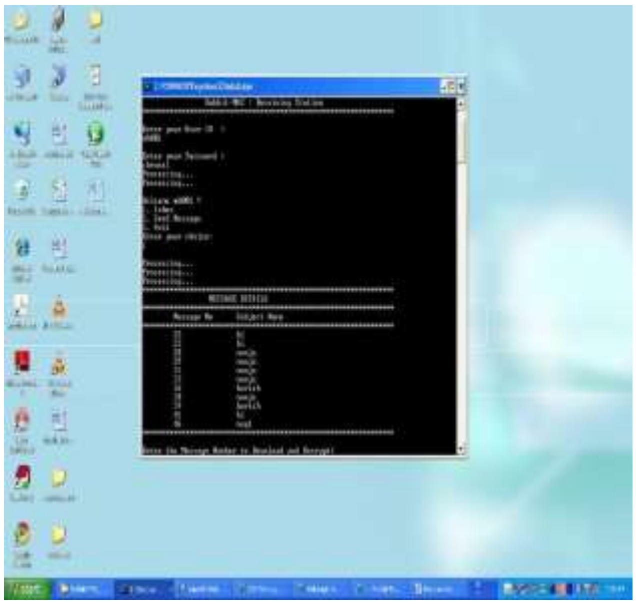
Send File

Client IP Address : 127.0.0.1

Message Name : manju

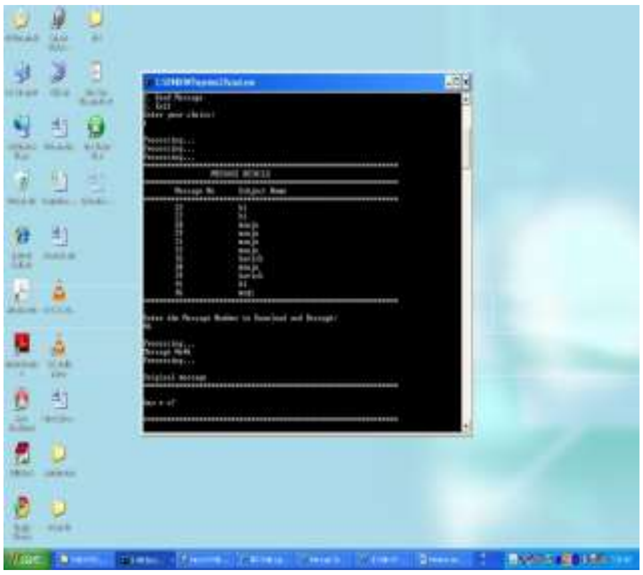
manju
hi
harish
msg1

Receiver login console and list of messages



```
            Rabbit-MAC : Receiving Station
==========================================================
Enter your User-ID :
[illegible]

Enter your Password :
[illegible]
Processing...
Processing...

Welcome [illegible] !
1. Inbox
2. Send Message
3. Exit
Enter your choice:
1

Processing...
Processing...
Processing...
==========================================================
               MESSAGE DETAILS
==========================================================
      Message No      Subject Name
==========================================================
      [illegible]     hi
      [illegible]     hi
      [illegible]     manju
      [illegible]     manju
      [illegible]     manju
      [illegible]     manju
      [illegible]     harish
      [illegible]     manju
      [illegible]     harish
      [illegible]     hi
      [illegible]     msg1
==========================================================

Enter the Message Number to Download and Decrypt:
```

Decryption Process



```
[illegible]
. Send Message
. Exit
Enter your choice:
[illegible]

Processing...
Processing...
Processing...
************************************************************
              MESSAGE [illegible]
************************************************************
      Message No      Subject Name
************************************************************
      [illegible]     hi
      [illegible]     hi
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     [illegible]
      [illegible]     hi
      [illegible]     [illegible]
************************************************************

Enter the Message Number to Download and Decrypt:
[illegible]

Processing...
[illegible]
Processing...

Original message
************************************************************

[illegible]

************************************************************
```

Receiver acknowledge the Message to sender

VI CONCLUSION

The developed system is highly interactive and user friendly . The performance of the system is found to be efficient and data maintenance and manipulation is achieved. The system has overcome all the problems in transferring the secret messages. Rabbit-MAC supports authenticated encryption, which provides message authentication and message integrity

It overcomes the Existing system limitations and Since it is a new stream cipher algorithm so No attacks against Rabbit have been published until now.

VII REFERENCES

[1] Ruhma Tahir, Muhammad Younas Javed 1, Ahmad Raza Cheema "Rabbit- MAC: Lightweight Authenticated Encryption in Wireless Sensor Networks", Proceedings of the 2008 IEEE International Conference on Information and Automation June 20-23 ,2008, Zhangjiajie, China

[2] J.-P. Aumasson. On a bias of Rabbit. In Proc. SASC 2007. available from http://www.ecrypt.eu.org/stream /papersdir /2007 /033.pdf.

[3] Chris Karlof, Naveen Sastry, David Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD, November 2004.

[4] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," Commun. ACM, vol. 47, no. 6, pp. 53–57, 2004.

[5] Chris Karlof and David Wagner, "Secure Routing Wireless Sensor Networks :Attacks and Countermeasures",IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[6] B. Schneier, "Applied Cryptography".

[7] Martin Boesgaard, Mette Vesterager, Thomas Christensen, Erik Zenner "The Stream Cipher Rabbit" ECRYPT Stream Cipher Project Report 2005/006.

[8] Elaine Shi and Adrian Perrig, "Designing Secure Sensor Networks",IEEE Wireless Communications, pp 38-43 December 2004

[9] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller and Mihail Sichitiu, "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes" Workshop on Wireless Sensor Networks and Applications (WSNA '03) with MobiCom'03, Sep 2003

[10] Deian Stefan ,"Hardware framework for the rabbit stream cipher" proceedings of the 5th international ACM conference on Information security and cryptology ,verlag Berlin, Heidgeberg , 2010.