Implementation of Hybrid Cryptography for Wireless Sensor Network in TinyOS

Kayalvizhi.R[#1], Vaidehi.V[#2]

[#]*Department of Electronics Engineering, Madras Institute of Technology, Anna University, Chennai-44, India.*

[1]kavikkayal@annauniv.edu, [2]vaidehi@annauniv.edu

***Abstract*— Wireless sensor networks have applications in many areas. Security is vital, where sensitive information is sensed and processed. In general the cluster based sensor network is easy to control or monitor the activities and also it increases the network life time. Hence protecting the network from the vulnerable is predominant using appropriate cryptographic algorithm. The security of crypto algorithms purely depends on the key which is used, where the key management includes key generation, exchange, and storage of secret keys. The proposed model consists of clusters which are formed after the sensor deployment. Each cluster has a cluster head (CH) and cluster members communicate with the base station via cluster heads. Cluster heads can communicate directly with each other and with the base station directly or through neighbouring nodes. Each node in the cluster will have a shared secret key to communicate with its cluster head. To generate that shared secret key, Elliptic Curve Diffie Hellman algorithm is used which outputs 160 bit shared secret key. It is converted into 128 bits using Message Digest (MD5) algorithm and the resultant key is given to AES for encryption. Nodes can use this key to encrypt the messages. This hybrid scheme is simulated using TOSSIM (TinyOS) simulator. The security provided by proposed scheme with smaller key size is equivalent to that provided by other asymmetric algorithms with larger key size. Hence this hybrid key management scheme can be employed in real time scenarios.**

Keywords- Security, Key management, ECC, WSN.

I. INTRODUCTION

Wireless sensor network (WSN) is defined as a collection of sensor nodes deployed to collect or gather specific data and about a phenomenon and process at the base station. WSN has applications in critical areas like battlefield surveillance and health care monitoring where critical data is sensed and communicated. Security is important in such areas. Generally in sensor networks, during transmission the security of data has to be ensured from inside and outside attackers. Security services such as encryption and authentication are required to prevent eavesdropping, alteration, and spoofing. Tiny nodes in sensor networks are limited by power and computational capability. Hence, security schemes for WSN have to account for these factors.

Part of security provision scheme is key management technique, in which the keys are created, stored, protected, exchanged, used and destroyed. Keys used for encryption and decryption can be either symmetric or asymmetric.

In Symmetric keying, the same key is used at both sides (Sender and receiver) whereas in asymmetric keying, private and public keys are used to ensure confidentiality. Public Key is used for encryption (sender side) while the private key is for decryption (receiver side). The disadvantage of symmetric keys are number of keys need to be stored are high and scalability is very difficult. To overcome the memory tradeoff the asymmetric mechanism can be adopted for WSN.

Some popular asymmetric key generation algorithms are RSA, Elgamal and Elliptic Curve cryptography. RSA is the most widely used algorithm. RSA algorithm needs key size of 1024 bits to provide an acceptable level of security, and with the advanced technology the Private Key can be factored out. Because of these problems, RSA algorithm cannot be used in critical applications like battlefield surveillance. Hence ECC algorithm is preferred as it is very secure owing to implementation of elliptic curve discrete logarithm problem (ECDLP). Table I compares the key size and its ratio between RSA and ECC algorithms.

TABLE I

COMPARISON OF KEY SIZE

ECC Key Size (bits)	RSA Key Size (bits)	Key Size Ratio
160	1024	1:6
224	2048	1:9
256	3072	1:12
384	7680	1:20
512	15360	1:30

II. RELATED WORK

A. Classification of keys.

Designing and implementing any kind of security mechanism requires a shared secret key (usually called the cryptographic key) to construct a trust relationship between two or more communicating parties. Managing these cryptographic keys play a vital role in providing reliable,

robust, and secure communication. Generally, five types of keys are used in a network. Figure 2.1 shows the classification of keys.

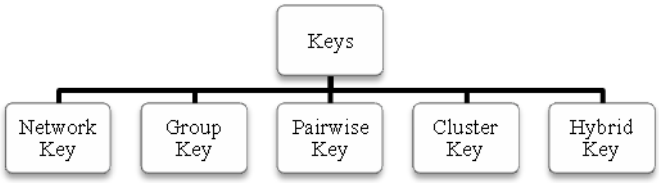

Figure 2.1: Types of keys

- Network key: A unique key shared with the base station.
- Pair-wise key**:** *S*hared with another sensor node.
- Cluster key**: S**hared with neighbouring nodes to secure local broadcasts.
- Group key**:** Shared by all the nodes in the network and is used to send broadcast messages.
- Hybrid key: A combination of any two or more of the above mentioned types.

B. Key Management Schemes

The key management schemes can be broadly categorized into three types as shown in figure 2.2,

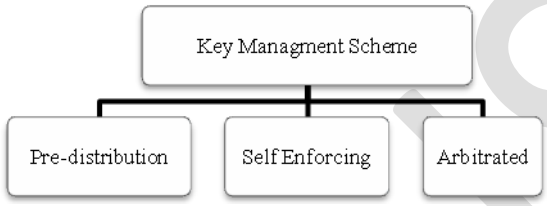

Figure 2.2: Key management schemes

a) Pre distribution keying mechanism: In this scheme sensors are pre loaded with set of keys [13]. It involves no computation, only selection of keys from the preloaded key pool. The following are the types of pre-distribution keying mechanism

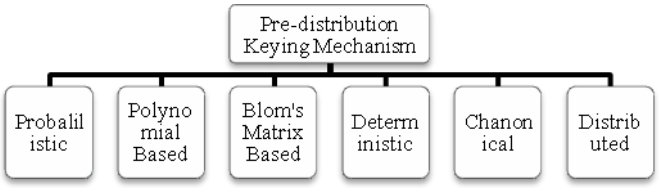

Figure 2.3: Pre-distribution keying mechanisms

Computation cost is low with pre distribution schemes. But such schemes are not very secure and require large memory to store large a key pool.

b) Self Enforcing keying mechanism: The self-enforcing scheme uses asymmetric cryptographic algorithms, which are limited by the current computation abilities and energy resources of sensor network technologies. The algorithms in asymmetric protocol are RSA and ECC algorithm. Asymmetric algorithm involves high computation [10] but provides high security.

c) Arbitrated Keying mechanism: The arbitrated keying scheme relies on some trusted central point or third party, which is vulnerable to single point of failure [13]. Authentication is ensured only if the third party is genuine. So it is difficult to find a trusted central point.

C. Existing Keying Schemes

In Eschenauer and Gligor scheme [6], each node has a key pool. When two nodes want to communicate, they will select a common key from the pool. Advantage of this scheme is that computations involved are lesser and simpler. However, there is no support for scalability and the memory requirement is high.

In Blom's model, private matrix of key is stored [5]. At the time of key establishment, only a single row and column are exchanged between two nodes. LEAP scheme [11] uses a pre-distributed key to establish four types of keys. It supports for cluster, pair-wise, and network-wide operations. But its storage cost is high for a small number of nodes due to the use of four types of keys.

SHELL scheme [7] generates and manages key using a distributed key management entity. Addition and replacement of nodes is possible with this scheme while the overall energy cost is higher due to complex operations.

Hence proposed method in this work deals with cluster formation, key creation, key exchange, encryption/decryption and rekeying processes. Static clusters are formed based on the region of coverage and cluster heads are fixed. Each cluster uses hybrid keying mechanism to establish shared secret key between cluster head and cluster nodes and is also used for encryption/decryption processes. Advantage with this scheme is, number of keys that every node has to store is very less.

III. IMPLEMENTATION

A. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key cryptography algorithm. The mathematical operations of ECC over the elliptic curve is defined in equation 1.

$$y^2 = x^3 + ax + b \quad (1)$$

where $4a^3 + 27b^2 \neq 0$. Each of the values of 'a' and 'b' gives different elliptic curve. All points (x, y) which satisfy the above equation, in addition to that a point in elliptic curve lies at infinite.

The random number is selected as a private key. This private key is multiplied with generator point (G) to produce the public key. The operations on elliptic curves are point addition, point multiplication and point inverse. The point multiplication is achieved through point addition and point doubling operations, whereas the point inverse is carried by tripling. The secrecy behind in the Elliptic curve cryptography is the discrete logarithm problem (ECDLP).

B. Proposed Hybrid Scheme

The hybrid approach mainly includes the following steps:

(i) Shared key generation using ECDH;
(ii) Key initialization;
(iii) Message encryption;
(iv) Decryption process.

a) Shared Key Generation

Initially the base station (BS) selects the elliptic curve and its private key and then computes its public key and sends it to all the nodes. Then, each node selects a random number as its private key and computes the public key and sends it back to BS. It also computes the shared key and stores it. BS, after receiving the public key from nodes, computes the shared key and stores it. Shared key is generated using ECDH algorithm by all nodes. Figure3.1 shows the key exchanging mechanism.

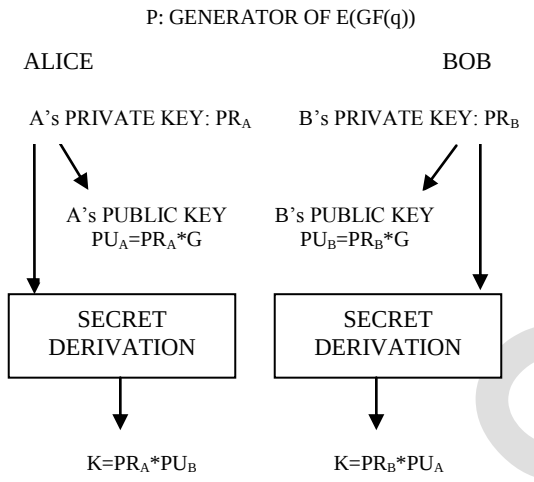

Figure 3.1: ECDH (Elliptic Curve Diffie Hellman)

Elliptic Curve Diffie-Hellman (ECDH) [4] is a key agreement protocol that allows two parties to establish a shared secret key over an insecure channel. Using the public data and their own private data the parties computes its shared secret key. Any third party, who doesn't have access to the private key details of each device, will not be able to calculate the shared secret key from the available public key information.

The generated 160-bit shared secret key is converted in to 128-bit key using MD5 such that it can be used by AES for Encryption and Decryption.

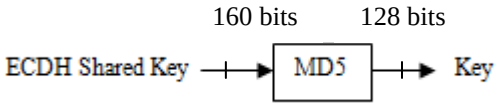

Figure 3.2: Key Generation using ECDH and MD5

b) Message Encryption

A node, say Alice encrypts her messages using AES algorithm. Figure 3.3 shows the block diagram for the encryption process. It takes the generated shared secret key (K) as an encryption key and transmits the cipher along with the message digest say M1 obtained by MD5 for the same input message.

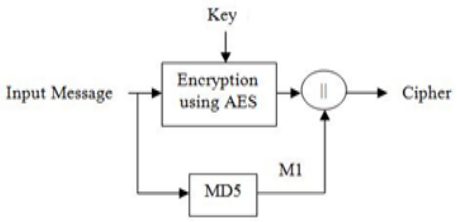

Figure 3.3: Block diagram of Encryption

The overall block diagram of shared key generation and encryption is shown in figure 3.4.

c) Decryption Process

At Reception, say Bob uses the shared secret key (K) to decrypt the message and gets the original message. The obtained plaintext is given as a input to the MD5 to get the message digest M2 and checks with the message digest received (M1). If M1 and M2 are same then it ensures the message's integrity. Thus Bob process the data else discards the data if an integrity of the message is lost.

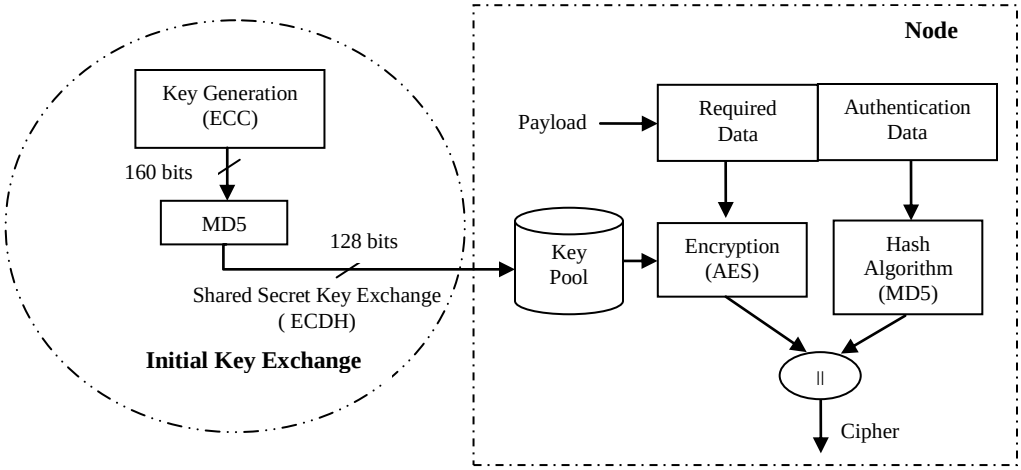

Figure 3.4: Proposed Hybrid Mechanism Block Diagram at Node

Suppose the receiver node is a cluster head (CH), once it gets the same input message from the same node id the generated message digest will be same. Then the CH checks for the redundant information then discards the received packet, which in turn reduce the communication cost.

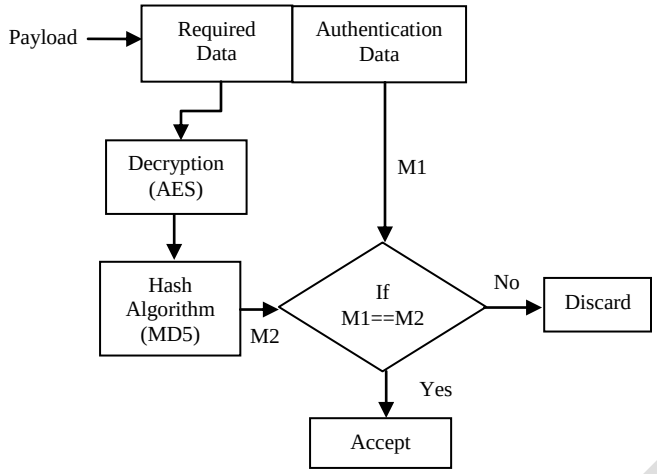

Figure 3.5: Proposed Hybrid Mechanism Block Diagram at Cluster Head / Base Station

C. *Implementation Step for the Proposed Scheme*

Key generation and Key exchange are based on ECC and ECDH algorithm respectively. The steps involved in the proposed scheme are as follows.

a) Installation of Tinyos: Hybrid key management scheme has been implemented in Tossim(TinyOS Simulator) based on nesC platform. Tinyos works in linux platform. To make it compatible with windows xp, can use cygwin.

- Download and install cygwin, tinyos 1.x package from the following link

http://webs.cs.berkeley.edu/tos/tinyos1.x/doc/install.html

- Once installation is complete, set environment variables properly as per the instructions specified in the above link.
- Use toscheck command to verify the tinyos installation. Only if toscheck is completed without errors, it will be possible to proceed with Tossim.
- To compile any program in Tossim use the command - make pc.
- To execute a program, use the command ./build/pc/main.exe n (n=45).
- To view the GUI, open new cygwin window, go to opt/tinyos1.x/tools/java/net/tinyos/sim and give the command tinyviz.

b) Node Deployment: Define the number of sensors to be created for the simulation scenario. The node placement could be random or in grid manner.

c) Cluster Formation: Once the sensors are deployed, clusters are formed in the network. Node with high computational capability becomes the cluster head. Nodes which are in the coverage range of cluster head will be included in the cluster as its member. Each node in the network will communicate with the cluster head and cluster heads will communicate with the base station. Cluster based communication will prolong the network lifetime as every node need not to communicate directly with base station. The static clustering methods aims at minimizing the total energy spent during the formation of the clusters for a given set of network parameters, such as the number of nodes in the network. Figure 3.6 shows cluster formation with node 14 as cluster head.

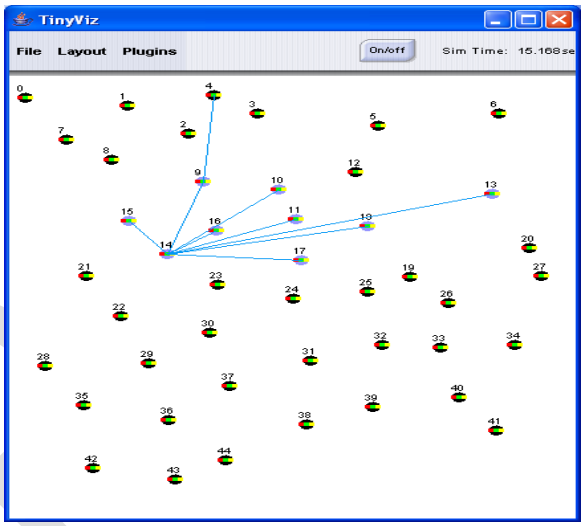

Figure 3.6: Cluster communication

d) Key generation using ECC:

Module EccM uses the interfaces specified below. SendMsg function is attached with debug message and send key message to send the information. Similarly Timer is also attached with debug timer and send timer for synchronization.

(Note: Ecc module can be found in the following link http://www.cs.harvard.edu/malan/projects/) [4].

```
module EccM
{   provides
  {
        interface StdControl;
  }

 uses
  {
    interface Leds;
    interface Random;
    interface ReceiveMsg;
    interface SendMsg as SendDbgMsg;
    interface SendMsg as SendKeyMsg;
    interface SysTime;
    interface Timer as DbgTimer;
    interface Timer as SendTimer;
  }
}
```

e) Key exchange using ECDH (Elliptic Curve Diffie Hellman): Cluster head sends the broadcast information along with its public key. After receiving the public key, the cluster nodes computes their shared secret keys using ECDH algorithm. Once the cluster member and their cluster head have a shared secret key, nodes can use that key to encrypt/decrypt the messages.

f) Symmetric key conversion: 160 bit shared secret key is changed to 128 bit digest using MD5 algorithm. This 128 bit key is used for encryption/decryption process.

g) Encryption: When an event has occurred, messages are encrypted using AES encryption algorithm and transmitted. As the keys are generated flowing ECDLP, security is ensured. Symmetric encryption algorithms involve lesser computation than asymmetric algorithms. Trace file of transmitted and received messages is shown in Figure 3.7.

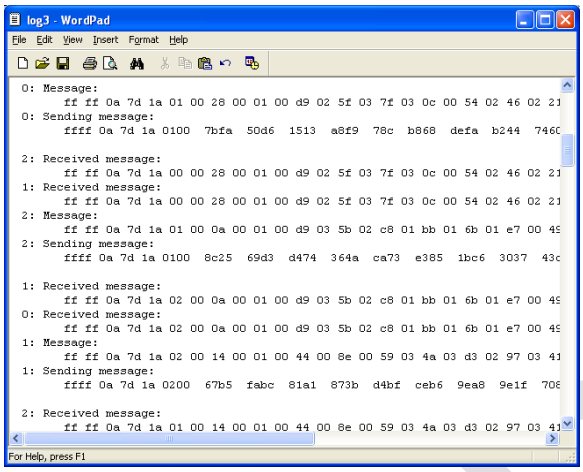

Figure 3.7: Trace file of simulation

The data packets are transmitted in the format shown in Figure 3.8. The fields indicate:

ff ff (2 bytes) - Broadcast Address
0a (1 byte) – Active Message ID
7d (1 byte) – Group ID
1a (1 byte) – Message Length
Remaining (29 bytes) – Payload

Destination address (2 bytes)	AM (1 byte)	Group ID (1 byte)	Data Length (1 byte)	Variable Length Payload (max 127 bytes)

Figure 3.8: TinyOS packet format

- The destination could be a broadcast address or unicast address. If it is broadcast the value is ff. If it is unicast, that node id is the value of destination address.
- Active message field could be 05 or 0a.
  05-Beacon message
  0a- Data packet
- Group Id-7d indicates default local group ID.
- Data length field indicates the length of payload transmitted.

For example, consider the node with node id 15. The packet generated by the node when the object is detected, is given below:

150000042**9070707070707070401**0000000000000*10380107252865792051601351678841*e599

where the bold portion represents the message for the data. The packet after encryption is given below:

150000042**935fe1d107a14e57ec041e6a32b1441bf**103801072528657920516013516788 41e599

where bold portion represents the cipher. The entire packet after decryption is given below:

150000042**9070707070707070401**0000000000000103801072528657920516013516788 41e599

where the bold portion refers the decrypted data packet.

IV. Results and Discussion

TABLE II

STORAGE REQUIREMENT

Algorithm	Number of keys to be stored
Symmetric Scheme	n(n-1)/2
Asymmetric Scheme	2n
Hybrid (proposed) Scheme	2n

Table II shows the required number of keys to be stored. Even though pair-wise keys are used between cluster head and cluster nodes, the number of keys stored is less compared to symmetric key management scheme because of hybrid keying mechanism which is a combination of symmetric and asymmetric keys; where n refers to the number of nodes.

Execution time and energy consumption of each algorithm is measured for a single node is shown in table III and IV respectively. RSA algorithm's execution time is 22 seconds. Even though RSA (1024 bits) execution time is lesser than ECC (160 bits), Key size used in the proposed scheme is also 160 bits which is much lesser than RSA 1024 bits. Hence the bandwidth requirement is reduced.

TABLE III

EXECUTION TIME

Algorithm	Time
ECC Key Generation	34 s
ECDH Key Exchange	3.1 s

AES Encryption	16 ms

ECC key generation will be executed only during the bootstrap process. Only the encryption process will be called often before sending the information. Hence the lifetime of the nodes will be enhanced by the proposed scheme. If a key is revealed by the hacker, there is a possibility of re-keying using ECDH algorithm.

TABLE IV

ENERGY CONSUMPTION

Algorithm	Energy consumption
ECC Key Generation	0.8160 J
ECDH Key Exchange	0.0744 J
AES Encryption	0.384 mJ

V. PERFORMANCE ANALYSIS

The most important operation in ECC is point multiplication. Several optimization algorithms exist, which reduces the time taken and energy consumed by the point multiplication. For instance, when Barrett reduction, hybrid multiplication and scalar multiplications are used. Table V consist of the memory requirement for implementation. The time taken for the various steps in key management and the energy incurred by the proposed method is shown in table VI and VII respectively.

TABLE V

MEMORY REQUIREMENT

Memory	Code utilisation	Available in Micaz
ROM	26458 bytes	128Kbytes
RAM	2679 bytes	4Kbytes

It concludes that the both execution time and energy consumed is low even though the code size increases. When optimization algorithms are included, there is a tradeoff between code size and execution time. For applications that execute point multiplication just once, there is no save in energy. But their always tradeoff between code size and the memory overhead.

TABLE VI

EXECUTION TIME

Process	Timing (Sec)
ECDH initialization	1.8546995
ECDH key generation	1.8125933
ECDH key agreement	2.1237543

TABLE VII

ENERGY CONSUMPTION

Algorithm	Energy consumption
ECC key generation	55 mJ
ECDH initialisation	44 mJ
ECDH key exchange	50.82 mJ
AES	38µJ

However, for applications that need to execute point multiplication algorithm multiples time, using sliding window code is advantageous. In this case, the increase in code size is traded off for the drastic reduction in execution time and thus energy is consumed.

CONCLUSION

Hybrid key management scheme has been simulated using TOSSIM Simulator. The AES algorithm consume less energy (38µJ), which is used frequently for the encryption and decryption process. Public Key Cryptography has not been used in the past due to the resource constraints on sensor platforms in wireless sensor networks. However after the successful implementation of EccM in sensor nodes, public key cryptography has gained popularity. ECC algorithm used for key generation is involved only in the initial stage. Though the complexity is high (150mJ), it provides better security. Thus, the proposed hybrid key management scheme can be employ in real time environments. This proposed method is also adoptable for cluster based communication, hence the network lifetime can be increased.

FUTURE WORK

Dynamic clusters could be used i.e. cluster head selection could be dynamic; it would be more efficient than the static clustering techniques. Key management ensures integrity and confidentiality; counter measures for possible attacks need to be tried.

REFERENCES

[1] An Liu, Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", International conference on Information Processing in sensor networks 2008.

[2] David J.Malan, Matt Welsh, Micheal D. Smith,"A Public-Key Infrastructure for Key Distribution in Tinyos Based on Elliptic Curve Cryptography", IEEE Sensor and Adhoc Communication Networks, pp.71-80. 2004.

[3] HailunTan, John Zic, Sanjay K. Jha, and Diethelm Ostry, "Secure Multihop Network Programming with Multiple One-Way Key Chains", Transactions on Mobile Computing, Vol. 10, January 2011.

[4] JIANG Jian-wei, LIU Jian-hui, "Research on Key Management Scheme for WSN based on Elliptic Curve Cryptosystem" Proceedings in IEEE Symposium on Security and Privacy, January 2009.

[5] Johnson c. Lee and Victor C. M. Leung, "Key Management Issues In Wireless Sensor Networks: Current Proposals And Future Developments", IEEE Wireless Communications, pp 76-84, October 2007.

[6] L. Eschenauer and V.D.Gligor, "A Key Management Scheme For Distributed Sensor Networks", in Proceedings of the 9the ACM conference on Computer and Communications Security, Washington DC, USA, November 18-22, pp. 41-47, 2002.

[7] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-Aware Combinatorial Key Management Scheme for Clustered Sensor Networks," IEEE Transaction on Parallel and Distributed Systems, vol. 17, pp. 865–82, 2006.

[8] M.Prabu, R.ShanmugaLakshmi, "A Comparative and Overview Analysis of Elliptic Curve Cryptography over Finite Fields", IEEE Conference On Information And Multimedia Technology, 2009.

[9] M. Yarvis, N. Kushalnagar, H. Singh, et al., "Exploiting Heterogeneity in Sensor Networks," Proceeding in IEEE INFOCOM 2005, Miami, FL, March 2005.

[10] Pritam Gajkumar Shah., Xu Huang, Dharmendra Sharma, "Sliding Window Method With Flexible Window Size For Scalar Multiplication On Wireless Sensor Network Nodes" IEEE Conference On Wireless Communication And Sensor Computing, 2010.

[11] Zhu S, S. Setia, and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," Proc. 10th ACM Conference on Computer and Communication Secience, pp. 62–72, 2003.

[12] Xiaojiang Du, Mohsen Guizani, Yang Xiao, "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks" Transactions on wireless communications, vol. 8, no. 3, March 2009.

[13] Yang Xiao, Venkata Krishna Rayi, Xiaojiang Du, Fei Hu, M.Galloway, "A Survey On Key Management Schemes In Wireless Sensor Networks", Computer Communications, Issue On Security on WSN, April 2007.