Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2015, 2(1): 1-7

Research Article ISSN: 2394 - 658X

Implementation of Secure Code Dissemination Protocol for Border Sensor Nodes

M B Nirmala and A S Manjunath

Department of Computer Science and Engineering, Siddaganga Institute of Technology, Karnataka, India
mbnirmala@sit.ac.in

ABSTRACT

Securing code dissemination is essential for many crucial military applications. Suppose the nodes are deployed in the military areas, the border sensor nodes are the more sensitive nodes. The border sensor nodes need to do some special tasks compared to other intermediate sensor nodes. So the code dissemination to be sent to the border nodes will be different from that of intermediate nodes. Thus, the code dissemination that needs to be sent to the border sensor nodes should be highly confidential. The confidentiality of this code dissemination cannot be compromised at any cost, hence Key management is also an important issue. A protocol is developed to securely send the code dissemination to the border sensor nodes via intermediate nodes. The Protocol is implemented using nesC on TinyOS platform and evaluated the performance.

Key words: Sensor node, Key management, Border nodes, TinyOS, Tossim simulator

INTRODUCTION

After the deployment of WSN, updating the program image for sensor nodes is subsequently essential to provide new functionalities and for bug fixes [1][3][14]. Security in wireless sensor networks has an increasing demand for monitoring application in military and civilian operations. Updating the program image securely is more important especially for military service applications [4][13]. For example, in environmental monitoring applications adversary should not gain any information about the new program image transmitted to sensor nodes since in wireless broadcast medium an attacker can easily inject or corrupt the packet. It is essential that we provide authenticity, confidentiality, data freshness, ordering for the code dissemination.

Suppose the nodes are deployed in the military areas (border areas) where the sensor nodes are the sensitive nodes. The border sensor nodes need to do some special tasks compared to other intermediate sensor nodes. So the code dissemination to the border nodes will be different from the other nodes. In our project we develop the protocol to securely send the code dissemination based on the location of the sensor nodes. The code dissemination that needs to be sent to the border sensor nodes is highly confidential. Since the dissemination need to be sent to the border sensor nodes over the network, there is a possibility that a malicious nodes try to modify the data being sent for their benefit. Main objective is to present an efficient, confidential network programming scheme for bordered wireless sensor networks. In our scheme, we follow distributed approach which consists of one base station, group heads and border sensor nodes. Base station divides the program image into packets and sends it to the group heads, which in turn sends the packets to sensor nodes.
Existing network reprogramming protocols employs the Deluge [2], Program image is divided into series of pages and a page into series of packets. To authenticate the sender of the code, hashing is applied on the last page as in [7] and hashed value is attached to the previous page. This process is applied recursively until the first page is found. Current research efforts on secure network programming protocols hash the last packet [6], then attach the hashed value to the second last packet. This process is applied recursively until the first packet is reached. After that, the first packet is signed with the private key of the base station. Each sensor node will have to verify the first packet with the public key and then authenticate each packet by the hashed value sent with the previous one. Deng

et al [4] proposed employing a Merkle tree instead of recursive hashing for packet authentication. These research efforts all employ digital signature to sign the first packet with the private key of the base station. There are two problems with this approach. First, signature verification can incur significant computation overhead, particularly power consumption in sensor nodes. Second, either sensor nodes are required to memorize the digital signature from the base station, or these schemes are constrained to one-hop program propagation because sensor nodes cannot generate a digital signature themselves as the base station for further propagation in multi-hop scenario. Lanigan et al[5] have some minor enhancements on Sluice by exploiting symmetric cryptography, but several issues have not been fully addressed for multi-hop network programming such as power consumption, multi-hop support and the impact of an adversary. All the above protocols provide authentication and integrity of the network programming, but none of them provide confidentiality. In confidential deluge [8] and [11] authors proposed a confidential and DOS resistant protocol using digital signature. But signature verification incurs computational overhead and additional power consumption in sensor nodes. Authors of [10] develop secure program dissemination protocol for clustered wireless sensor networks. This scheme uses Broadcast encryption [9] and provides confidentiality, authentication, and integrity and also is resilient to node compromise attack.

NETWORK MODEL ASSUMPTION

In our proposed system, we are assuming a distributed Wireless Sensor Networks with one Base station communicating with the Border Sensor Nodes. The base station is of higher computation ability and rich in resource compared to border nodes. Border nodes have higher computation abilities and rich in resource compared to other sensor nodes which are resource constrained. The base station sends the information to the intermediate nodes. These intermediate nodes route the messages to the group heads. Group-head in turn disseminates the information to the other border sensor nodes. Border nodes are more vulnerable to malicious attacks.

SECURE CODE DISSEMINATION PROTOCOL

This paper considers a set S of N sensor nodes $s_1, s_2, \ldots, s_N$. This set contains intermediate nodes, as well as border nodes. The border nodes are divided into several multicast groups and for each group there is a group head. A secret key is also generated among the border sensor nodes and group head. Shortest path is calculated from base station to group heads via intermediate nodes. Through these intermediate nodes, the packets are routed from the base station to the group heads. . The program image is divided into packets, each containing suitable header and trailer. The group head further decrypts the packet, encrypts them again using the secret key and sends them to the border nodes in its group.

Procedure for Secure Code Dissemination

Initialization: Base station, group heads and Sensor nodes are stored with a pre deployment key.

1. Secure group key is constructed among the dynamically constructed group key using group key protocol [12].
2. Base station sends the ADV message to all the nodes in network.
3. Sensor nodes willing to receive the code dissemination sends the REQ message back to sensor nodes.
4. While receiving the REQ message base station also records the shortest path information from the sensor nodes.
5. Base station computes the hash using the pre deployment key.
6. Base station prepares the packet and unicast it to one hop intermediate node.
7. Intermediate node in turn forwards it to next hop and the same continues until the group head is reached.
8. Once the packet reaches the group head, the group head decrypts the key, encrypts with group key and broadcast to the border nodes.
9. The border nodes after receiving the nodes, decrypts then and compute the hash on the packet with the help of a predeployed key.
10. If the computed hash and the received hash are same then the packet is stored otherwise discarded and sends the NACK message to the Base station. Base station will resend the packet.
11. Border nodes can construct the program image once all the packets are received.

SIMULATION DETAILS

The scheme is implemented in Tiny OS platform [15]. The code is written in NesC language and is tested on TOSSIM simulator. In this scheme we have a grid of sensor motes which are distributed randomly, where we have one base station, a few intermediate nodes and border nodes. All the border nodes are three to four hop distance away from the base station. The number of border nodes depends on the total number of nodes. Suppose we have N nodes, at least N/4 nodes are considered as the border nodes. All the border nodes are divided into several multicast groups depending on the location. Each multicast group has a group head. 0^{th} sensor node is always considered to be the trusted Base station. The border nodes are grouped based on the consecutiveness of their node Ids. Hence every group's starting node is considered as the group head. For example: Suppose N=40, then network consists of 0th node as base station. Figure 1 shows the sensor node deployment and Figure 2 shows an example of formation of multicast group and selection of group head. Set of node IDs {23, 10, 26, 23, 36} form one multicast group, where, node ID 23 being the group head. Similarly set of nodes IDs {7, 4, 8, 19, 12, 16} form multicast group with 7 being the group head and set of node IDs {15, 6, 1, 13, 20, 28} with 15 as the group head. Figure 3 shows the transmission of packets to group head over intermediate nodes. Figure 4 shows the transmission of packets from group head to border nodes.

In this section the discussion on the code blocks for finding the shortest path, various communication messages sent from base station to intermediate nodes and from intermediate nodes to sensor nodes.

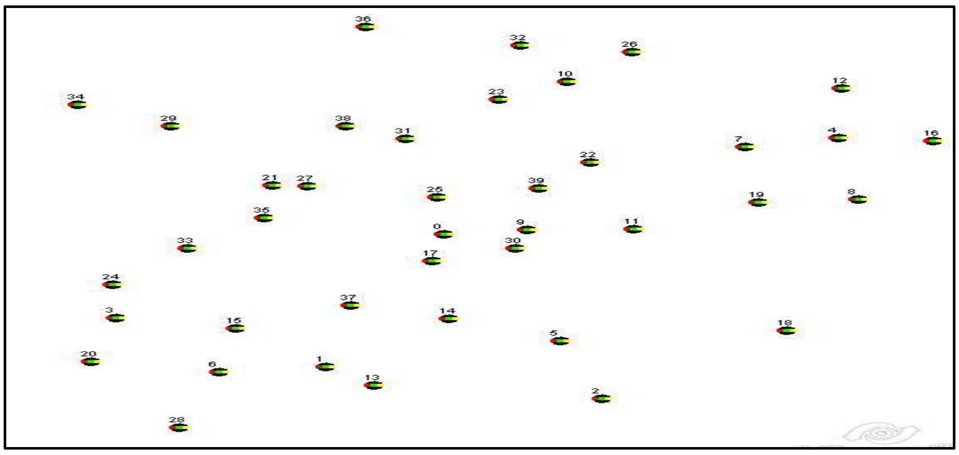

Fig. 1 Sensor nodes deployment

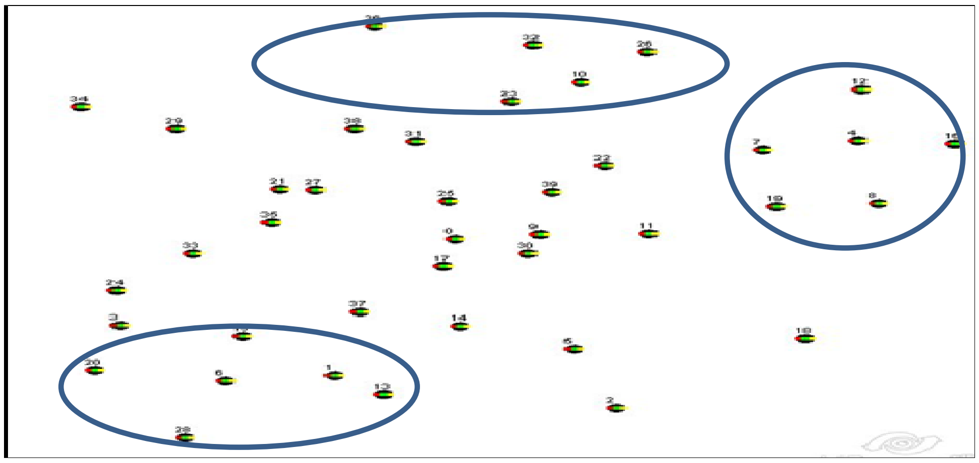

Fig. 2 Example of multicast groups and group head

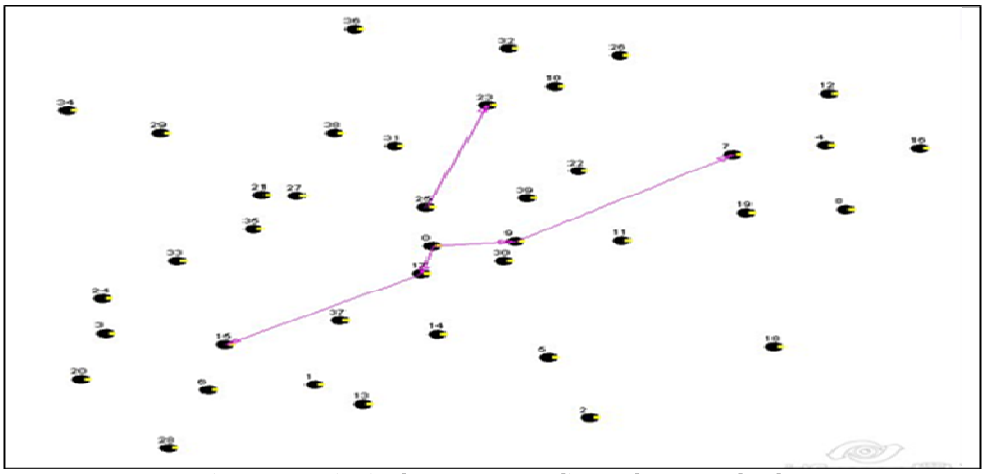

Fig.3 Communication between Intermediate node to group head

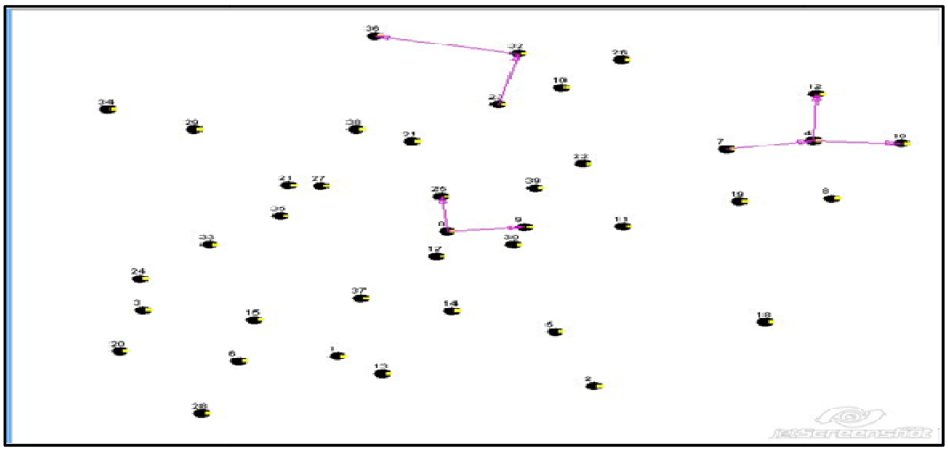

Fig.4 Communication between group head to border nodes

Configuration

The following is the configuration of our component.

```
Configuration Ring
{
}
implementation
{
            components Main, RingM, TimerC, GenericComm as Comm, LedsC;
            Main.StdControl -> RingM.StdControl;
            Main.StdControl -> Comm;
            Main.StdControl -> TimerC;
           RingM.Timer -> TimerC.Timer[unique("Timer")];
            RingM.SendMsg -> Comm.SendMsg[AM_SIMPLEMSG];
            RingM.ReceiveMsg -> Comm.ReceiveMsg[AM_SIMPLEMSG];
            RingM.Leds -> LedsC;
}
```

Module

Some of the existing modules and interface of TinyOS used in the implementation of the scheme is as follows:

```
module RingM
{
            provides interface StdControl;
            uses interface SendMsg;
            uses interface ReceiveMsg;
            uses interface Timer;
            uses interface Leds;
}
implementation
{
//Here code is written for key generation, key computation, code dissemination,    Sending and receiving the data.
}
```

Packet Format

Base station has the binary program image. The size of the program image can be varied. This program image is divided into packets as per the deluge protocol. In our implementation each packet consists of 48 bytes data. The packet format is shown below.

The first byte of the packet is dedicated to indicate the address of the sender that is source address. For example: 0 in case of base station. Second byte is reserved for the sequence number of the packet. Length of the actual data in the packet is stored at the 64th byte. 62nd byte consists of the index value of the key from the subset of the key pool associated with that particular destination. 63rd byte consists of the node ID of the intended destination. Sequence number, index value, destination and the length are not encrypted since, at the receiver before decrypting the data the node need to, retrieve the key, verify the source, destination and also check for duplication of the packet. 3rd byte to 50th byte is used for storing the encrypted data. From 51st byte to 61st byte encrypted Hashed value of the data is stored. In the implementation, while sending above packets are copied to the AM sampling message of type 50.

Table -1 Packet Format

Source address	Sequence number	Encrypted data	Encrypted hash value	Index value of the key	Intended Destination node ID	Length
1byte	1byte	48bytes	11bytes	1byte	1byte	1byte

Integrity Authentication and Confidentiality

In order to enforce error control and check the integrity of the packet, SHA1 is applied on the payload of each packet. . SHA1 takes variable length input with maximum length of less than 264 bits and gives an output of 160 bits message digest. In this scheme as we use only 48 bytes data or payload so only first 11 byte of hash value is of concern. After computing the hash it is stored from 51st byte to 61st byte of the sending packets. Hashing is done at base station. At the receiving side both group head and border sensor nodes apply SHA1 to compute hash on the received data and compared.

For purpose of providing authentication and confidentiality RC5 32/12/16 is used for encryption and decryption. 32 bit block of data is encrypted at a time using 16 8 bit keys and 12 rounds of operation. The output is 32 bits cipher texts, stored at 3rd byte to 63rd byte. Two rounds of Encryption one for 32 byte of data and another for rest 16 byte of data along with 13 byte hashed value with appended zeros are performed. RC5 involves Key expansion algorithm for enhancing the 16 keys to 36 keys. Two keys are used in each round of encryption with other two used for addition operation not a part of the encryption.

Shortest Path Calculation

In this phase a simple ADV packet is broadcasted to all the nodes in the network by the base station. The time taken to receive the REQ packet from each node is determined by base and stored. These variables are compared and the node which receives in shortest time is decided to be the first intermediate node in the path. Then this node again broadcasts a ADV packet to other nodes in the network to determine the next node in the path. The following code shows how shortest time is calculated

```
void mst()
{       t1=clock();
        if((call SendMsg.send(TOS_BCAST_ADDR,  sizeof(msgc.data), &msgc)) != SUCCESS)
        {
                dbg(DBG_USR1, "SEND MSG FAIL\n");
        }
}
 event TOS_MsgPtr ReceiveMsg.receive(TOS_MsgPtr m)
{       t2=clock();
        node[TOS_LOCAL_ADDRESS].val=t2-t1;
        // further code
}
event result_t Timer.fired()
{       min= node[1].val;
        for(i=2;i<NODE1;i++)
        {               if(i%4!=0)
                {
                        if(node[i].val<min)
                        { min=node[i].val; minNode=i; }
                }
        }
// further code
}
```

Communication

The TinyOS provides the interface called GenericComm for communication between the nodes. This interface provides the function for sending and receiving the date. General format is

```
   SendMsg.send(destinationID,sizeofData,pointerToData);
```

- destinationID : ID of the receiver
- sizeofData : Size of the actual data to be sent in bytes.
- pointerToData: Pointer to the buffer where data is copied for sending.

Base station broadcasts the ADV packets to all the nodes using following code

```
void mst()
```

```
{
t1=clock();
if((call SendMsg.send(TOS_BCAST_ADDR, sizeof(msgc.data), &msgc))!= SUCCESS)
          {
dbg(DBG_USR1, "SEND MSG FAIL\n");
          }
}
```

Base station sends the encrypted messages to the intermediate nodes using the following code:
msgb is a variable of TOS_MSG which contains the encrypted data. nextminNode contains the next intermediate node in the shortest path. Group head sends the message to all nodes in its multicast group using the following code:

```
task void sendkeyb()
          {          if(ci<numpack*numnodes1)
         {
           if((call SendMsg.send(dest1, sizeof(msgb.data), &msgb))!=SUCCESS)
          {
                    dbg(DBG_USR1, "SEND  MSG FAIL\n");
          }
          else {node[dest1].flag=1; ci++; dest1=dest1+4; }
          }
  }
```

PERFORMANCE ANALYSIS

To evaluate the performance Sensor nodes are deployed varying from 10 to 70 nodes and the program size is considered from 5 KB to 25 KB. Figure 5 gives the time taken to transmit the program image with varying size from 5 KB to 25 KB. This shows that as program size increases the end to end latency also increases. Figure 6 shows the time required for the transfer of the image of size 10 KB and 20 KB from base station to the Border Sensor nodes, with varying number of nodes since increase in number of nodes increases the number of hops in network.

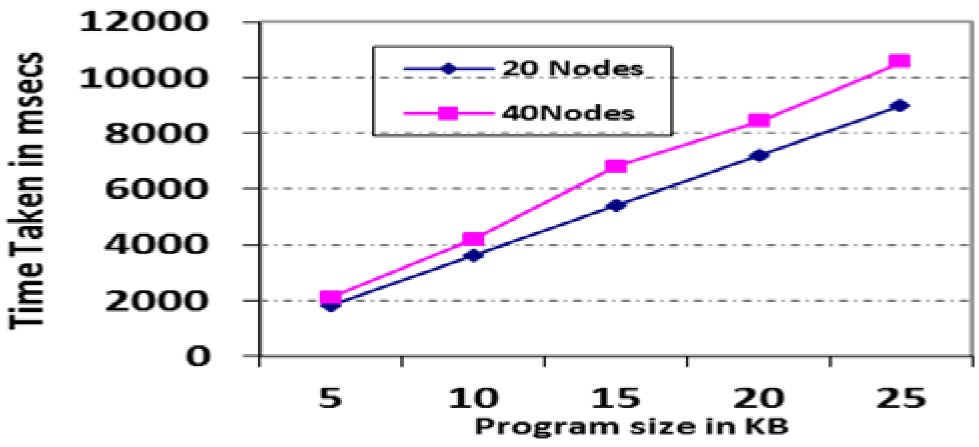

Fig.5 Time Taken vs program size in KB

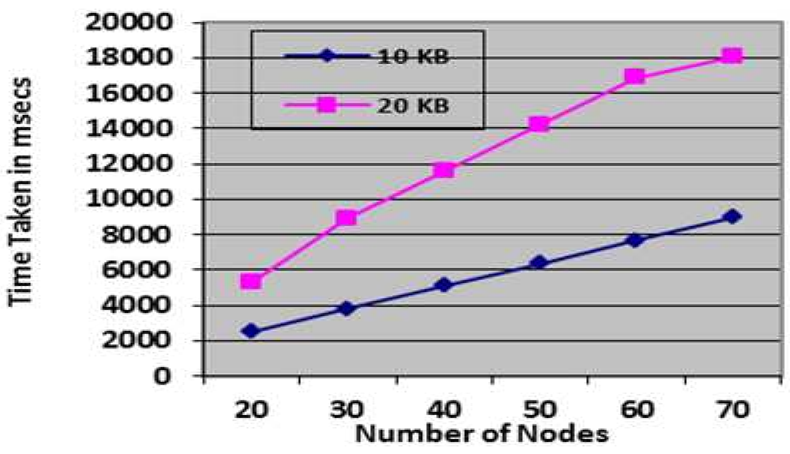

Fig.6 Time taken vs number of nodes

CONCLUSION

This paper deals with secure program dissemination for distributed wireless sensor networks with multi-hopping and multicasting. Border sensor nodes are considered to be four to five hops distance away from the base station. Confidentiality and integrity is achieved. Performance is evaluated for different program size and number of nodes.

REFERENCES

[1] I F Akyildiz, Weilian Su,Y Sankarasubramaniam, and E Cayirci, A survey on Sensor Networks *Communications Magazine, IEEE,* **2002,** 40(8), p.102-114 .

[2] Jonathan W Hui and David Culler, The Dynamic Behaviour of a Data Protocol for Network Programming at Scale, *SenSys '04*, New York, USA, **2004**, *ACM press*, p. 81-94.

[3] Limin Wang, MNP: Mulltihop Network Reprogramming Service for Sensor Networks, *SenSys '04*, *ACM press*, **2004**, p. 285-286.

[4] Jing Deng, Richard Han and Shivakant Mishra Secure Code Distribution in Dynamically Programmable Wireless Sensor Networks *IPSN '06*, New York, USA, *ACM press*, **2006**, p. 292-300.

[5] P E Lanigan, R Gandhi, and P Narasimhan, Sluice: Secure Dissemination of Code Updates in Sensor Networks, *26th IEEE International Conference on Distributed Computing Systems (ICDCS '06),* **2006**, p.53-63.

[6] Prabal K Dutta, Jonathan W Hui, David C Chu, and David E Cullar Securing the Deluge Network Programming System, *IPSN '06*, New York, USA, **2006** *ACM Press*, p. 326-333.

[7] Hailun Tan, Sanjay Jha, Diet Ostry, John Zic, and Vijay Sivaraman, Secure Multi-hop Network Programming with Multiple One-Way Key Chains, *WiSec '08: Proceedings of the first ACM conference on Wireless Network Security, ,* New York, USA, **2008**, ACM, p. 183-193.

[8] Hailun Tan, Diethelm Ostry, John Zic and Sanjay Jha A Confidential and DOS-Resistant Multi-hop code Dissemination Protocol for Wireless Sensor Networks, *Computers and security*, **2013,**32, p. 36-55.

[9] Nirmala M B and A S Manjunath, Secure Program Update using Broadcast Encryption for Clustered Wireless Sensor Networks, *6th IEEE Conference Wireless Communications and Sensor Networks, IIIT-Allahabad, UP, India,* **2010**.

[10] Nirmala M B, A S Manjunath and Yogesh B G, Tiny SPU: Implementation and Analysis of Secure Program Update for Clustered Wireless Sensor Networks, *Global Trends in Computing and Communication System, Communications in Computer and Information Science, Springer*, **2012,** Vol. 269, pp 703-713.

[11] Hailun Tan, Diethelm Ostry, John Zic and Sanjay Jha A Confidentail and DOS-Resistant Multi-hop code Dissemination Protocol for Wireless Sensor Networks, *Wisec '09*, Switzerland, **2004**, *ACM press*, p. 245-252.

[12] R Dutta and T Dowling, Secure and Efficient Group Key Agreements for Cluster Based Network, *Transactions on Computational Science Iv: Special Issue on Security in Computing, Lecture Notes In Computer Science*, **2009,** vol. 5430, p. 87-116.

[13] Dennis K Nilsson, Roosta, Ulf Lindqvist and Alfonso Valdes, Key Management and Secure Software Updates in Wireless Process Control Environments, *WiSec '08: Proceedings of the First ACM conference on Wireless network security,* New York, USA, **2008**, p. 100-108.

[14] T Y Chi, W C Wang, S Kuo and Yu Flow Dynamic Software Updating in Wireless Sensor Networks, Proceedings of the 8th International Conference on Ubiquitous Intelligence and Computing, Banff, AB, Canada, **2011**, p. 405-419.

[15] P Levis, S Madden, J Polastre, R Szewczyk, K Whitehouse, A Woo, D Gay, J Hill, M Welsh, E Brewer and D Culler, *TinyOS: An Operating System for Sensor Networks, www.cs.berkeley.edu/~culler/papers/aitinyos. pdf.*