IJCAT International Journal of Computing and Technology, Volume 1, Issue 6, July 2014
ISSN : 2348 - 6090
www.IJCAT.org

Key Retrieval Mechanism for Data Aggregation in IPKE Managed WSN

[1] Sheetal Pawar, [2] Dr. J. W. Bakal

[1] ME Student, Department of Computer Engineering, MGM's College of Engineering & Technology
Kamothe, Navi Mumbai, University of Mumbai, India

[2] Principal, S. S. Jondhale College of Engineering & Technology
Dombivali, Mumbai, University of Mumbai, India

Abstract - Establishment of pairwise keys in wireless sensor networks is a fundamental security service, which forms the basis of other security services such as authentication and encryption. However, due to the resource constraints on sensor nodes, establishing pairwise keys in wireless sensor networks is not a trivial task. For the existing key pre-distribution schemes, as the number of compromised nodes increasing, the fraction of affected pairwise keys will increase quickly. As a result, a small number of compromised nodes may affect a large fraction of pairwise. The main purpose of providing security for node is to send data to sink node securely. In existing scheme IPKE node to node delivery is secured but if any node is compromised then data passing through it is also visible to attacker. This problem can be solved if a tunnel is formed between source node and sink node. In this case no intermediate node can interpret data as it is encrypted with the key shared by source node and sink node.

***Keywords* - Key pre-distribution, Pairwise keys, Wireless sensor network, Improved Pairwise Key Establishment (IPKE), Key Distribution Center (KDC)**

1. Introduction

A wireless sensor network is composed of a large number of sensor nodes for covering wider area through multi-hop connections, and has various kind of application including environmental monitoring, industrial monitoring, safety and security services, military system, health-care services, etc. These mission critical applications for wireless sensor networks make security and privacy functions required. However, To achieves security in wireless sensor networks is a challenging task, particular due to the constrained capabilities of smart sensor nodes (battery supply, CPU, memory etc.) and the harsh deployment of a sensor network. Secure, keys for performing encryption and authentication must be agreed upon by the communication nodes. However, due to the resource constrains on the sensor nodes, many key agreement mechanisms used in general networks, such as Diffie-Hellman and other public-key based schemes , are not feasible in sensor networks.

Currently there are three types of key management schemes that have been studied in wireless sensor networks: trusted server scheme, self-enforcing scheme, and key pre-distribution scheme. A trusted server scheme depends on a trusted server for key distribution and management. This type of scheme is not very suitable for wireless sensor networks because there is usually a lack of a trusted infrastructure in the application environments in which wireless sensor networks are used. Self-enforcing schemes, on the other hand, relies on asymmetric cryptography, e.g., key distribution and management using public key certificates. However, limited computation and energy resource in sensor nodes usually make it undesirable to use public key algorithm, such as RSA, for the sake of energy conservation. The third type of key management scheme, i.e., the key pre-distribution scheme, is such a scheme in which key information is pre-distributed among all sensor nodes prior to deployment. Such schemes seem most appropriate for wireless sensor networks, and it is type of scheme we consider here.

Two straightforward solutions can distribute symmetric keys into wireless sensor nodes. The first solution is to let all the sensor nodes store an identical master secret key. Any pair of nodes can use this global master secret key to achieve key agreement and obtain a new pairwise key. Although this approach looks very simple and efficient, this scheme does not exhibit desirable network resiliency. If one node is compromised, the security of the entire wireless sensor network will be compromised. Some existing studies suggest storing the master key in temperresistant hardware to reduce the risk, but this increase the cost and energy consumption of each sensor node. Furthermore, tamper-resistant hardware might not always to safe. At the other extreme, one might consider, give each pair of sensor nodes a distinct pairwise key, which means each sensor node needs to store (*n*-1) different pairwise key in its memory if there *n* nodes in a wireless sensor network. This solution has the perfect network security since any sensor node's capture or compromise cannot affect the communication between non-compromised nodes. The main limitation of this solution is the key storage overhead, which makes it not suitable for large-scale wireless sensor networks.

Most proposed schemes are either based on the probability and random graph theories, or based on bi-variate polynomial calculations.

To address the limitation of existing key pre-distribution schemes, we propose an improved pairwise key establishment scheme for wireless sensor networks in this paper. Compared with existing approaches, our scheme is secure against node capture attack. Low storage overhead, complete network connectivity, large network size, low communication and computational overhead are other benefit of our scheme.

The remainder of this paper is organized as follows: In Section 2, we discuss the weaknesses of existing key pre-distribution schemes. A detailed description of our proposed improved key retrieval mechanism in IPKE is presented in Section 3. Section 4 gives the security analysis and performance evaluation. Section 5 summaries our work.

2. Related Work

In this section we present overview of various significant concept proposed in literature. Few recent proposals are as follows:

Eschenauer and Gligor [1], proposed first key pre-distribution scheme in which sensor nodes are assigned a random subset of keys from a large key pool before deployment of the network. After deployment, two neighboring sensor nodes can establish a pairwise key between them as long as they have at least one common key in their key rings. In this approach, a very large size symmetric key pool is generated offline first. Each sensor randomly selects a set of keys from the generated key pool as its pre-distributed keys. Then the sensors are randomly deployed into an interested terrain to execute the corresponding operation. After the deployment, each node broadcasts its stored key information to its one-hop neighbors. Since all the keys are randomly selected from the same key pool, it is quite possible that two neighboring nodes have some overlapped keys. If two sensors have a common key, they can use it as their pairwise key directly. Otherwise, a path-key establishment procedure is triggered, which could generate a path-key between the two communicating nodes under some other intermediate node's participation.

Chan, Perrig, and Song [2] proposed a *q*-composite random key pre-distribution scheme, which focuses on the security of key setup such that an attacker has to compromise many more nodes to achieve a high probability of compromising communication. The difference between the q-composite scheme and the scheme in [1] is that this scheme requires at least *q* (*q*>1), instead of just a single one shared keys for two sensor nodes, to establish a shared key. These *q* keys are hashed into one key to achieve better resiliency to sensor node capture. The number of required shared keys makes it exponentially harder for the attacker to compromise a link key with a given subset of already compromised keys.

Both of above schemes cannot guarantee the entire network's connectivity with one-hop neighboring node's key information. To achieve appropriate network connectivity, intermediate nodes are requested to generate path-key between two communicating nodes, which not only degrades the network security, but also produces additional communication and computational overheads. Another weakness of previous schemes is the scalability. Due to the heavy key storage overhead of each sensor, these schemes cannot be used for a large-scale WSN. Although Chan et al.'s scheme improves the network resilience when the total number of the captured nodes is low; its performance degrades dramatically when the number of the captured nodes exceeds a critical value.

Blom [4] proposed a mechanism to setup a pairwise key between any two members in a group. In this approach, a $(\lambda -1)\times n$ matrix *G* and a $(\lambda -1) \times (\lambda -1)$ symmetric matrix *D* are constructed first, where *n* is the group size and λ is the expected threshold of how many members can compromise the secret collusively. Each member randomly selects a row vector from matrix *A* , $A = (GT * D)$, and a corresponding column vector form matrix *G* . If two members want to communicate each other, they exchange their column vectors first. Then, each side multiplies its row vectors with its partner's column vector. Due to the property of symmetric matrix, the two members can get a same number and use it as their pairwise key. Blom's scheme is perfectly secure when the number of the compromised members is less than λ; but once more than λ members are compromised, all the secret information of the group would be broken. This property is called "λ – se*curity"*, which is the main limitation of Blom's approach.

To improve security two random key spaces schemes [4, 7] have been proposed. Du et al. [3] apply Blom's key pre-distribution mechanism [5] for shared key establishment in sensor networks. Liu et al. [4] proposed a similar pairwise key scheme based on Blundo's polynomial-based key distribution scheme [6]. In both schemes, a number of key spaces are pre-computed and each sensor is associated with one or more key spaces before deployment. Two sensor nodes can compute a pairwise key after deployment if they have keying information from a common key space. In these two schemes, the communication between non-compromised sensor nodes keeps secure when the number of compromised sensor nodes is less than a critical value. But once the critical value is exceeded, the adversary would crack all the pairwise keys.

To achieves better scalability, a deployment knowledge base key management method is proposed by W.Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili [7]. They proposed multiple deployment points which are identified in the sensor network and for each deployment point; a key space is pr-computed. Neighboring deployment points have a number of keys in common. All sensor nodes are grouped before deployment and each group corresponds to one deployment point. Each sensor randomly picks several keys from the key space of its group. After deployment, sensor nodes in close neighborhood have a high probability of sharing a common key. This scheme has strong requirements on deployment, but achieves better scalability compared with those proposed in [3][4][5][6]. A general frame for establishing pairwise keys in wireless sensor networks is studied in [8], which is based on the polynomial-based key pre-distribution protocol proposed by [7].

Y. Cheng, D. P. Agrawal [9] proposed an efficient pairwise key establishment scheme, which focuses on four phases to established a pairwise communication key. These phases are: setup key pre-assignment phase, common keys discovery phase, pairwise key computation phase, and key ring establishment phase. In this scheme, a large-size two-dimensional key matrix is constructed to distribute setup keys to sensors. Each sensor randomly selects a row and a column from the matrix before deployment. Since each pair of row and column has an intersection, this scheme ensures any pair of sensors shares at least two common keys. A pairwise key is generated by the common keys and communicating parties' ids. This scheme not only has lower storage and communication overheads than previous schemes, but also can achieve complete network connectivity no matter how the sensors are deployed. The weakness of [9] is it is vulnerable to node impersonate attack. Since node's id is broadcasted in the initialization phase, a malicious node may catch that information and impersonate a good node to carry out attacks.

To address the limitations of previous schemes, Y. Cheng, D. P. Agrawal proposed an improved pairwise key establishment scheme[10], which has better performance in terms of network resilience, connectivity, communication overhead and memory storage.

There is a severity flaw in most existing key pre-distribution schemes, which an attacker can get key information of the uncompromised nodes from the compromised nodes.

3. Key Retrieval Mechanism in IPKE

This scheme provides security against node capture attack in wireless sensor network. Because in existing key pre-distribution schemes, an attacker can get key information of the uncompromised nodes from the compromised nodes. To alleviate this issue, the proposed scheme uses tunneling to improve security against node capture attack. This security is achieved for data gathering purpose in the proposed work. So, data travels toward SINK node securely as it forms a tunnel between SINK and child node. So that, no other sensor node can decrypt the data.

In proposed scheme, two keys are used one is setup key and other is communication pairwise key. Setup key is pre-loaded into sensors and used to establish a secure link between sensors. The communication pairwise key is established by two communicating parties using their shared setup keys and some random numbers they generate under certain rules. Only the generated pairwise keys are used to encrypt/decrypt the communication between sensors. Each pairwise key is distinct to others, any sensor's capture cannot compromise non-captured nodes pairwise key.

3.1 Two Phases of IPKE

Here, a pair wise key is established through two phases, setup key pre-loading phase and a pairwise key generation phase.

3.1.1 Setup Key Pre-loading Phase

In this phase, KDC (Key Distribution Center) selects a set of keys under certain rules from key pool P and pre-loads them into each sensor node to ensure any two nodes share at least two common keys after the deployment. To achieve this requirement, a setup key preloading procedure is executed.

First, KDC randomly selects n keys (where, n is the expected number of sensors in a WSN) from the key pool P, and uses these keys to construct a (m×m) key matrix K (where m = |sqrt (n)|). Fig.1 is an example of a constructed key matrix K , where each key has a unique two-dimensional id denoted as $kc_{i,j}$ where (i, j = 1,2,...,m) . We use kr_i , where (i = 1,2,...,m) and kc_j , where (j = 1,2,...,m) to represent the i^{th} row and the j^{th} column of the matrix K , respectively.

$$\begin{array}{c|cccccccc|}
 & kc_1 & kc_2 & kc_3 & kc_4 & kc_5 & kc_6 & .. & kc_m \\
\hline
kr_1 & k_{1,1} & k_{1,2} & k_{1,3} & k_{1,4} & k_{1,5} & k_{1,6} & .. & k_{1,m} \\
kr_2 & k_{2,1} & k_{2,2} & k_{2,3} & k_{2,4} & k_{2,5} & k_{2,6} & .. & k_{2,m} \\
kr_3 & k_{3,1} & k_{3,2} & k_{3,3} & k_{3,4} & k_{3,5} & k_{3,6} & .. & k_{3,m} \\
kr_4 & k_{4,1} & k_{4,2} & k_{4,3} & k_{4,4} & k_{4,5} & k_{4,6} & .. & k_{4,m} \\
kr_5 & k_{5,1} & k_{5,2} & k_{5,3} & k_{5,4} & k_{5,5} & k_{5,6} & .. & k_{5,m} \\
kr_6 & k_{6,1} & k_{6,2} & k_{6,3} & k_{6,4} & k_{6,5} & k_{6,6} & .. & k_{6,m} \\
: & : & : & : & : & : & : & .. & : \\
kr_m & k_{m,1} & k_{m,2} & k_{m,3} & k_{m,4} & k_{m,5} & k_{m,6} & .. & k_{m,m} \\
\hline
\end{array}$$

Fig. 1 A constructed setup key matrix K.

For each sensor node, KDC randomly selects a row and a column from K and pre-loads these keys into the particular sensor to consist its key ring. Figure 3 illustrates the key rings stored in nodes a and b, where a stores 1st row and 2nd column of matrix K, and b stores 4th row and 6th column of matrix K. This setup key pre-loading procedure ensures any two sensors share at least two common keys.

For instance, it is easy to see that nodes a and b have $k_{1,6}$ and $k_{4,2}$ *in* common. Since each pair of nodes shares at least two common key, IPKE guarantees any two sensors to establish a secure link between them after the deployment. Therefore, our scheme can provide full secure network connectivity no matter how and where the sensors are deployed lately, which releases the assumption of prior knowledge of sensor deployment location.

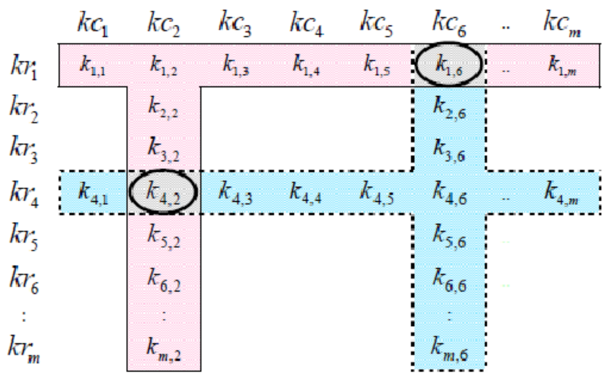
kc_1 kc_2 kc_3 kc_4 kc_5 kc_6 .. kc_m
kr_1 kr_2 kr_3 kr_4 kr_5 kr_6 : kr_m
$k_{1,1}$ $k_{1,2}$ $k_{1,3}$ $k_{1,4}$ $k_{1,5}$ $k_{1,6}$ .. $k_{1,m}$
$k_{2,2}$ $k_{3,2}$ $k_{4,2}$ $k_{5,2}$ $k_{6,2}$: $k_{m,2}$
$k_{2,6}$ $k_{3,6}$ $k_{4,6}$ $k_{5,6}$ $k_{6,6}$: $k_{m,6}$
$k_{4,1}$ $k_{4,3}$ $k_{4,4}$ $k_{4,5}$ .. $k_{4,m}$

Fig. 2 Shared common keys between two sensor nodes.

3.1.2 Pair Wise Key Generation Phase

After deployment, any two neighboring sensors need to generate a pairwise key between them. To do that, each node needs to find the shared common keys with its neighbors first.

Suppose sensors a and b are two neighboring nodes. Node a first broadcasts a message { N_a , kr_1 , kc_2 , $nonce_a$ } to its one-hop neighbors, where N_a is the id of node a , kr_1 is the index of the key row stored in a , kc_2 is the index of key column stored in a , and a *nonce* is a one-time used random bit-string which is generated by a . (Here, $nonce_a$ is used to prevent the adversary to derive the pairwise key directly from a compromised key ring lately). Similarly, node b broadcasts a message { N_b , kr_4 , kc_6 , $nonce_b$ } to its neighbors too.

After exchanging the broadcasting information, node a obtains $nonce_b$, kr_4 and kc6 from node b. According to (kr_4 , kc_2) and (kr_1 , kc_6), node a can figure out that it shares keys { $k_{1,6}$, $k_{4,2}$ } with node b . Similarly, node b can get the corresponding information from node a. Once the key information exchanging is finished, node a and node b can calculate their pairwise key by Equation (1).

$$PK_{a\text{-}b} = nonce_a \;\square\; k_{(a,\,b)} \;\square\; nonce_b \qquad (1)$$

where "□" is the exclusive-or operator, $PK_{a\text{-}b}$ denotes the pairwise key between nodes a and b , $k_{(a,\,b)}$ are the common keys shared between a and b (in this example they are $k_{1,6}$ and $k_{4,2}$).

After the pairwise key generation phase, each sensor erases all the pre-loaded setup keys from its memory to prevent the possible compromise in the future. Now, we can see, by randomly pre-distributing a row and a column from the key matrix K into each sensor, any pair of sensors can share at least two common keys in their memories, which means, any two sensors within their radio transmission range can directly establish a secure link between them without the third node's involvement. In addition, each side of the communicating parties generates a random nonce to participate the pairwise key generation procedure, which prevents the adversary to compose the established pairwise key lately even it can compromise the stored key rings in sensors.

3.2 Tunneling

The proposed work uses tunneling to improve security against node capture attack. The security is achieved for data gathering purpose. So, data travels toward sink node securely as it forms a tunnel between sink and child node. So that, no other sensor node can decrypt the data. Here, sink generates the key pool and distribute sink key to each node. If a child node want to send a data to sink. First, it encrypts the data with sink key then adds message digest which is calculated using CRC algorithm. Then again encrypts data using its parent's pairwise key. After that, parent node can encrypt data using sink key and sending to sink. After receiving data, sink decrypts data using parent's pairwise key and then checks message digest, if it is same then again decrypts data. So, all the intermediate nodes cannot read that data which is delivered from node to sink. Only sink node can decrypt the data. So, this forms a tunnel between node and sink. It also provides security from the compromised node.

4. Implementation Details

The proposed system uses Improved IPKE (Improved Pairwise Key Establishment) scheme which is based on random key pre-distribution techniques. The goal of this scheme is perfect security and energy efficiency. It uses tunneling to improve security against node capture attack. The security is achieved in proposed scheme for data gathering purpose. So, data travels toward sink node securely as it forms a tunnel between sink and child node. So that, no other sensor node can decrypt the data. Here we are using CRC16 algorithm for calculating message digest.

The algorithm details are as below.

Algorithm 1- For key establishment with adjusting node:

```
1. Start
2. Construct m x m key matrix(M) and install on
   each node
3. For each node repeat
    a. Send { $N_a$ , $kr_a$ , $kc_a$ , $nonce_a$ } to neighbour
    b. Collect { $N_b$ , $kr_b$ , $kc_b$ , $nonce_b$ } from
       neighbour
    c. Key1=M[$kr_a$ $kc_b$]; Key2=M[$kr_b$ $kc_a$]
    d. $k_{(a, b)}$ = Key1 $\oplus$ Key2
    e. $PK_{a\text{-}b} = nonce_a \oplus k_{(a, b)} \oplus nonce_b$
4. Stop.
```

Algorithm 2- For key establishment with Sink node:

```
1. Start
2. Construct m x m key matrix(M) and install on
   each
   node
3. For each node repeat
    a. Send { $N_a$ , $kr_a$ , $kc_a$ , $nonce_a$ } to neighbour
    b. Collect { $N_{sink}$ , $kr_{sink}$, $kc_{sink}$ , $nonce_{sink}$ }
       from neighbour
    c. Key1=M[$kr_a$ $kc_{sink}$]; Key2=M[$kr_{sink}$ $kc_a$]
    d. $k$ = Key1 $\oplus$ Key2
    e. $PK_{sink} = nonce_a \oplus k \oplus nonce_b$
4. Stop
```

Algorithm 3- For sending data to Sink node:

```
1. Start
2. Repeat for all nodes till current node is not Sink
       node
    a. Append PK$_{sink}$ to own data and Calculate
       CRC
    b. Append CRC to own data and Encrypt with
       sink key PK$_{sink}$
    c. Append own encrypted data to data received
       from child node
    d. Encrypt data with the key PK of the parent
       node
    e. Forward data to parent node
3. Decrypt data received sequentially in the order
   they are encrypted with
4. Stop.
```

Algorithm: Node to Sink data Transfer(Tunneling):

```
Void Node_to_Sink_Encrypt( char * data, unsigned short
length)
{
    //Calculate hash for data
      unsigned short CRC = crc16(data,length);
    //Use IPKE to find key that to be used for encryption
      KEY=IPKE_establish_key(Node);
    //Now place hash code in the original data such that
      even if attacker able to decrypt data it should not
      able to retrieve hash code and change it.

new_length=Pack_CRC_with_data(data,CRC,length);

     encrypt(data,new_length,KEY);
```

}

5. Simulation

IpkeAgent class Definition:

```
class IpkeAgent : public Agent {
public:

        char *keysr;        //Key row
        char *keysc;        //Key Column
        int kr,kc;          //Row column numbres
        char nonce;
        int maxNodes;
        char child_keys[100];       //
        int child[100];
        double     iEnergy;
        MobileNode  *iNode;
        int node_no;
        char parent_key;
        int parent;
        int child_count;
        char sink_key;
        IpkeAgent();
        void selectKeys();
        short unsigned crc16(char*, short unsigned int);
        int command(int argc, const char*const* argv);
        void recv(Packet* p, Handler*);
        void ipkeTimeout();
        class IpkeTimer : public TimerHandler
        {
                public:
                IpkeTimer(IpkeAgent &a) : agent(a) { }
                IpkeAgent &agent;
        } ipkeTimer;

   };
```

Above code shows definition IpkeAgent class. This class implements the IPKE protocol. Both sender and receiver policies are defined here.

1. selectKeys(): This function randomly selects key from Key matrix and stores them in keypool of node.
2. Crc16(): This function implements CRC16 function
   and returns its CRC for data to be sent.
3. Command(): Following commands are implemented
   in this protocol.
   a. Negotiate: This command used to initialize keys for nodes and sink.
   b. Set_index: This sets index for each node
   c. Set_parent: This command assigns parent to a node
   d. Set_child: This command sets children to parent nodes
   e. send_data_without_encryption:This command sends data to sink without applying any encryption and crc.
   f. send_data_without_digest: This command sends data with encryption but without adding crc code.
   g. Send_data: This sends data to with encryption and crc code.
4. Recv(): This function processes the received packet.

6. Results and Analysis

6.1. Energy Consumption for “With Encryption” and “Without Encryption”

Fig. 3 shows remaining energy in node after sending packets. And Fig. 4 shows energy consumed by the node. Comparison shows that energy consumed by the node in both cases is similar. There is not much difference in the consumptions. This proves that our encryption algorithm is energy efficient.

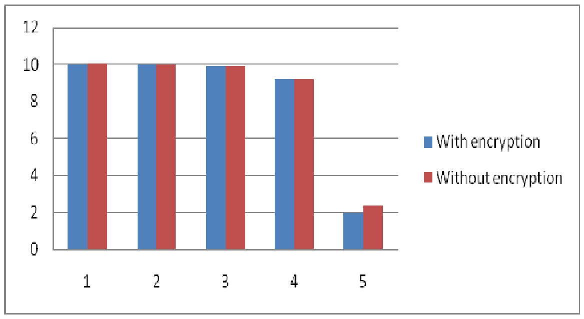

Fig.3 Energy remained graph

Table 1: Energy remained table

No. of nodes	With encryption	Without encryption
1	9.986789	10
10	9.977426	9.978665
100	9.905306	9.909585
1000	9.183897	9.218789
10000	1.98064	2.32356

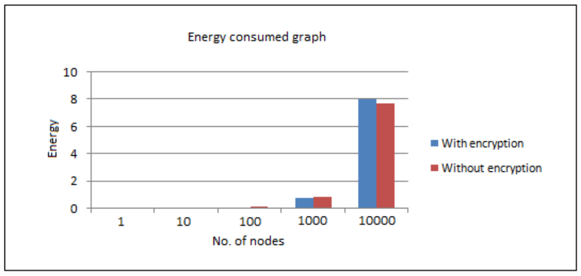

Fig. 4 Energy consumption graph

Table 2: Energy consumption table

No. of nodes	With encryption	Without encryption
1	0.013211	0.002
10	0.022574	0.021335
100	0.094694	0.090415
1000	0.816103	0.781211
10000	8.019036	7.67644

6.2 Energy Status of Idle Node and Busy Node

An idle node is a node which is not sending any packet to sink where as busy node is a node which is sending encrypted packet to sink. Fig. 5 and table 3 shows consumption of energy with respect to time in ms. It shows that after 1000ms the busy node has sufficient energy remained. The idle node is also losing energy as some energy is consumed by system software of the node.

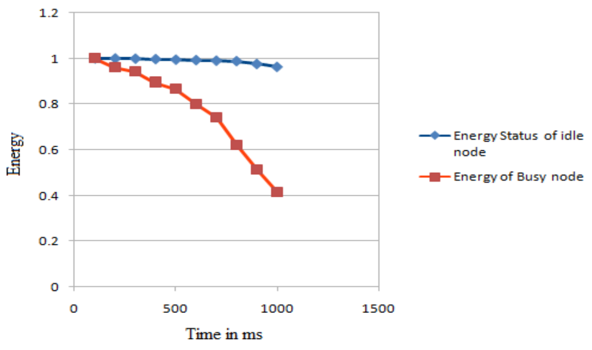

Fig.5 Energy status graph for idle and busy node

Table3: Energy status values for idle and busy node

Time(ms)	Energy Status of idle node	Energy of Busy node
100	0.99999	0.9994
200	0.99991	0.961
300	0.9991	0.9426
400	0.997	0.8942
500	0.9949	0.8658
600	0.9928	0.7974
700	0.9907	0.739
800	0.9886	0.6206
900	0.9765	0.5122
1000	0.9644	0.4138

6.3 Energy Consumptions for Different Packet Sizes

Here node is sending 1000 packets of each of the differnet sizes of the packets. After sending packets of size 1200 continuosly lot of energy is utilised. Whereas if packet size is less then utlisation is very less. So it is recommened that instead of sending big packets small chunks of packets are more energy efficient.

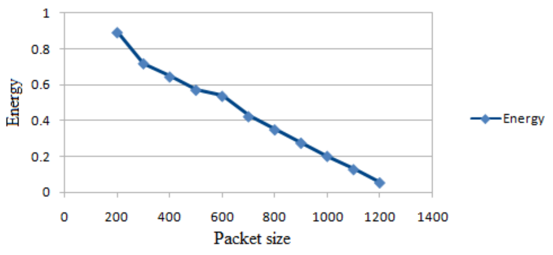

Fig.6 Energy consumption graph for different packet sizes

Table 4: Energy consumption values for different packet sizes

Packet Size	Energy
200	0.8942
300	0.72
400	0.6458
500	0.5716
600	0.5374
700	0.4232
800	0.349
900	0.2748
1000	0.2006
1100	0.1264
1200	0.0522

7. Conclusions

This paper proposed an improved key management scheme for wireless sensor networks that some compromised sensor nodes only affect part of uncompromised sensor nodes. With the one-way hash function, the proposed scheme can make attackers get less key information from the compromised sensor nodes. Performance evaluation results show that the proposed protocol has low memory and communication overhead and compared to existing key pre-distribution schemes, the proposed scheme is substantially more resiliency against sensor nodes capture.

In this, tunnel is formed between sink node and source node so that no intermediate node can predict the data to be forwarded. Even if the intermediate node is compromised the security is confirmed atleast for the node which is not compromised. It is proved from the results the energy required for sending data from node to sink is very less. Before sending data each node has to encrypt data with sink key and then with its own key. Because of dual key the authenticity and integrity is always ensured. Results are taken for different size of the packet but it is always seen that energy required is always in acceptable range. Node to node delivery is also secured.

8. Future Scope

This protocol can be revised to add some security features. WSN and data aggregation techniques can be improved further to ensure extra security. As depth of node tree increases delivery time also increases. For a big network, clustering can be done and multiple sinks can be added. As memory size of sensors is always limited key matrix cannot store keys for large network in such a case network clusters are needed. Although the techniques prove that security in WSN is improved but one cannot guaranty that it cannot be compromised. If any using this protocol sensor nodes can't be protected from physical attacks.

References

[1] L. Eschenaure and V.D. Gligor, "A key-management scheme for distributed sensor Networks". in: Proc. of the 9the ACM Conference on Computer and Communications, Washington DC, USA, pp.41-47, Nov. 2002

[2] H. Chan, A. Perrig and D. Song, "Random key predistribution schemes for sensor networks", in: Proc. 20003 IEEE Symposium on Security and Privacy, pp.197-313, May 2003

[3] W.Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili, "A pairwise key System predistribution schemes for sensor networks networks". ACM Transactions on Information and Security, Vol.8. No2, May (2005)228-258

[4] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks". ACM Transactions on Information and System Security, vol.8, pp.41-77, Feb. 2005

[5] R. Blom, "An optimal class of symmetric key generation systems. Advance in Cryptography" London, UK: Springer-Verlag, pp.335-338, 1985

[6] C. Blundo, A. D. Santis, A. Herzberg. S. Jutten, U. Vaccaro, and M. Yung. "Perfectly secure key distribution for dynamic conference", Information and Computation, vol.1, pp.1- 23, Jan. 1995

[7] W.Du, J. Deng, Y.S. Han, P.K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution schemes for sensor networks using deployment knowledge".IEEE INFOCOM pp597, 2004

[8] D.Liu and P.Ning, "Location-Based pairwise key establishment for static sensor networks", Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, pp.72-82, 2003

[9] Wenqi Yu, "A Pairwise Key Management Scheme Based on Hash Function for Wireless Sensor Networks", 2010 Second International Workshop on Education Technology and Computer Science, IEEE, 2010.

[10] Po-Jen Chuang, Tun-Hao Chao, and Bo-Yi Li, "A Scalable Grouping Random Key Predistribution Scheme for Large Scale Distributed Sensor Networks", Third International Conference on Information Technology and Applications (ICITA'05), IEEE, 2005

[11] Y. Cheng, D. P. Agrawal, Efficient Pairwise Key Establishment and Management in Static Wireless Sensor Networks, in: Proc. of the 2nd IEEE International Conference on Mobile Ad-Hoc and Sensor Systems, Washington, DC, Nov. 7-10, 2005.

[12] Y. Cheng, D. P. Agrawal, Improved Pairwise Key Establishment for Wireless Sensor Networks, in: Proc. of the IEEE International Conference on Mobile Computing, Networking and Communications, 2006, pp 442~448.