International Journal of Communication Networks and Information Security (IJCNIS) Vol. 4, No. 1, April 2012

Multiple Secret Keys based Security for Wireless Sensor Networks

V. Thiruppathy Kesavan[1], S. Radhakrishnan[2]

Department of Computer Science and Engineering
Kalasalingam University, Tamil Nadu, India

E-mail: { [1]kesavan, [2]srk }@klu.ac.in

Abstract- We propose a security approach that uses secret key cryptography and key management along with re-keying support. A salient feature of our approach is that a secret key is embedded in the source code of every node to protect the other keys in its non-volatile memory. Even the node is captured physically; the sensitive information cannot be retrieved. Our key selection protocol uses the node ID and some basic rotate and multiplication function to select the key for current data transmission. Because of this dynamic key selection, our approach identifies the replay attack, DoS attack and Sybil attack. Our simulation results shows that our security mechanism efficiently controls various attacks with lower resource requirements and the network resilience against node capture is substantially improved.

Keywords**:** secret key, re-keying, replay, DoS, Sybil, WSN

1. Introduction

Modern advancements in wireless technology have enabled the growth of packed in, low-power, multifunctional wireless sensor nodes that look smaller in size and can communicate in short distance even in un-tethered environment. Collections of these wireless sensor nodes form a dynamic, multi-hop, routing network connecting each sensor node to more powerful traditional networks and processing resources. In the battlefield surveillance application, sensor nodes could monitor the passage of vehicles and some times used to track the position of enemy or even safeguard the equipment [1]. Some other critical applications like forest fire detection [2], the wireless sensor networks are designed for early detection of forest fires.

Wireless Sensor Networks (WSNs) application such as military application has mission-critical tasks and so it is clear that security requirement to be taken into account during the design time itself. Furthermore, most of the WSN should run continuously and reliably without any interruption. Hence incorporating security in wireless sensor networks is very challenging.

Sensor Node consists of both volatile and non-volatile memory. In the non-volatile memory the static information such as program, node-ID, routing table, and security related data can be stored. Due to the improvements in the hardware technology, the physical size of memory is reduced by increasing the capacity of memory.

WSNs are vulnerable to various types of attacks that include jamming attack [3], eavesdropping, packet replay attack, modification or spoofing of packets, node replication attack, Sybil attack, flooding attack, wormhole attack, sinkhole attack, denial-of-service (DoS) attacks, node compromise attack and injection of false messages through compromised nodes [4][5][6].

The key distribution and management are considered to be the core of secure communication. In our proposed security mechanism, the keys are not directly distributed over the network at any time. Instead, the parameters that are used to generate the keys are transmitted only during re-keying. It is significantly hard for an adversary to identify those parameters.

In the remainder of this paper, we address the related works in section 2 and describe our proposed work in section 3. Section 4 describes the storage requirement for our security approach. Section 5 states the security analysis on various attacks. Section 6 presents the comparison of our mechanism with other security solutions. Section 7 provides the performance evaluation based on simulation. Section 8 concludes this paper and outlines further work.

2. Related Works

One of the points to be noticed is that no key distribution scheme is ideal to all kind of sensor network applications [7]. If the key is distributed during the lifetime of the network, it may be modified or hacked by the adversaries. So it is better to generate the keys for every round or session as and when required. Chin-Ling Chen and Cheng-Ta Li [9] have proposed a dynamic key management mechanism for WSNs, where the keys that are required in the next round are generated dynamically using the previous two keys that are already preset in each sensor nodes. They are using a one-way hash function for generating the new key.

The sensor nodes should not always depend on static keys that are preloaded or generated only once during its entire lifetime. It should have the re-keying facility to revoke the keys that are identified by the adversaries or lifetime of the key is expired [10]. Pietro *et al.*, [8] proposed a protocol called KeEs which is composed of a key generation and a key distribution/ synchronization phases. During the key generation phase, a key is involuntarily generated by every sensor node in the network in a time-triggered approach The KeEs protocol considered the major security key establishment protocol properties such as session key secrecy, forward secrecy and backward secrecy. Since the authors assumed the possibility of chosen plain text attack, they have employed periodic re-keying in order to reduce the cipher text availability to the adversaries.

Many key distribution and management schemes are proposed such as pair-wise shared key schemes [11][12][13]

and random key pre-distribution scheme [9][14]. Sharing a single secret key among all the nodes is vulnerable to attack. Instead every node can have different pair-wise keys is much more secure, but this solution occupies unnecessary storage space on a sensor node [15]. Instead of pre-distributing the keys in each sensor nodes, some parameters can be pre-distributed and that parameters can be used to dynamically generate the keys [9].

Security architectures have been proposed by considering the design issues of WSN. SenSec [16] uses a variant of skipjack algorithm called skipjack-X for generating the cipher text by introducing one more secret key without affecting the internal structure of the algorithm. SenSec does not defend against replay attack. Kalpana Sharma et. al. [17] introduced Intelligent Security Agent architecture that uses trust framework which consists of 11 parameters to compute trust level of all its neighbors. It requires more amount of memory to maintain these parameters.

SPINS [18] is a security framework that does not focus on implementation efficiency, instead they focused mainly on security protocol. But TinySec [19] architecture focused on implementation efficiency. In this architecture, key scheduling has to be pre-computed in RC5 which requires additional 104 bytes of RAM per key.

Kui Ren et. al. [20] proposed a location-aware-end-to-end security framework which is robust against DoS attack. It uses efficient en-route false data filtering scheme in order to identify the false data injection attack. This framework uses the preloaded master key along with its cell's location to generate the cell key by hash operation. The major drawback in LEDS is its increased resource consumption due to hop-by-hop authentication, hop-by-hop decryption, processing and encryption. To deliver an event to the sink, it broadcasts the event message that leads to consume more amount of energy. The number of keys maintained in every node depends upon the number of endorsements T. Hence the key storage overhead is directly proportional to T.

Table I. Notations

Symbol	Explanation
BS	Base Station
SK	Secret Key
DK	Data Encryption Key
RK	Re-keying Key
DK_i	i^{th} Key to be used for encrypting the data
$EDKn_i$	Encryption of data using DK of node n_i
$ERKn_i$	Encryption of data using RK of node n_i
C	Counter value that maintains the number of bits to be rotated
RKRQ	Re-keying Request
RKA1	Re-keying Authentication *Message 1* from BS
RKA2	Re-keying Authentication *Message 2* from node to BS
RKPM	Re-keying Parameters send from BS
RKPA	Re-keying Parameters received acknowledgement from node to BS
RDT	Random Text
SoD	Sum of Digits

3. Proposed Security Approach

In this section, we present the overall details of our security approach that ensures the following security properties:

Backward Secrecy: Even if an adversary recovered an adjacent subset of keys, it is impossible to recover the previous keys.

Privacy: Even the node is physically captured by an adversary; the secret information in the node's memory cannot be retrieved.

Data Integrity: Data Integrity ensures that the data during transmission over the network is not modified by an adversary.

Secure Management: Our mechanism provide secure method for key generation as well as for re-keying which is very much necessary in defending against cryptography attacks [4].

Our approach has three types of keys:

Data Encryption Keys (DKs): keys that are generated and shared within a group and BS.

Re-keying Key (RK): key that is generated and shared between a node and BS which is used during re-keying.

Secret Key (SK): key that is shared between a node and BS.

The keys DK and RK were encrypted using SK and maintained in its volatile memory. Due to this little bit of computational overhead, even if the nodes are physically captured, the keys cannot be retrieved from its volatile memory.

3.1 Network Assumptions

Our approach assume wireless sensor network in which the nodes are static with similar computational and communication capabilities. The network uses skipjack algorithm for encryption and decryption process. We have chosen this algorithm because the memory requirement is very less and encryption/decryption and key setup efficiency is also good [21].

To have variations in having the keys, we have used logical grouping of nodes for maintaining different set of keys. In a group, all the nodes maintain same set of keys, but every node uses different key for different communications with the base station.

3.2 Design Goals

Our proposed approach is designed to identify the DoS attack, Packet Replay attack and Sybil attack. Identifying those attacks will help to increase the network lifetime.

3.3 Grouping of Nodes

If all the nodes in the network are using same set of keys, all the nodes have to participate in re-keying which is an overhead. To reduce this overhead, the nodes are grouped based on the size of the network. After grouping, if any one node needs re-keying, the other nodes in that group itself have to participate in re-keying process. This avoids the overhead of re-keying for the remaining nodes which belongs to other group(s). Let the number of nodes be N, types of key be N_K, number of groups be N_G and number of data encryption keys N_{DK} per node is limited to 9. Let us take this example.

- N = 100 (Number of nodes in the network)
- N_K = 2 (Data encryption keys and Re-keying keys)
- N_{DK} = 9
- N_G = Round ((N/N_K)/ 9) = 6 groups

An example network with 6 groups of keys is shown in Fig. 1.

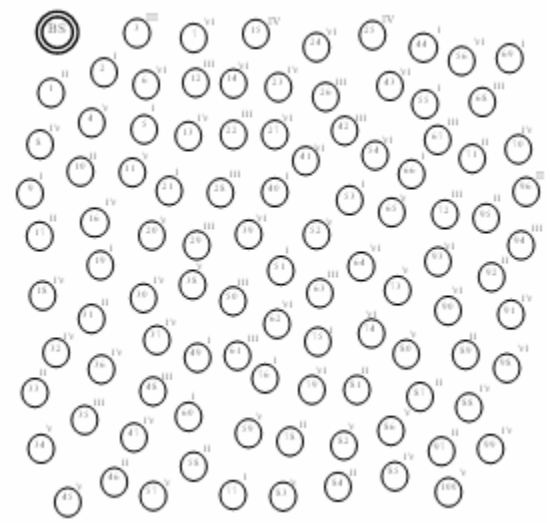

Figure 1. Network with six groups of keys

3.4 NODE DEPLOYMENT

Before a node is deployed, the static Secret Key (*SK*) has to be embedded in the source code and convert the same to its executable (*.exe*) format and loaded in the node's non-volatile memory. Then every node is pre-distributed with 2 pairs of parameters, say (k_i, k_{i-1}) and (r_i, r_{i-1}) which are used for generating Data Encryption keys and Re-keying keys respectively using one way hash function. A unique seed value $Seed_i$ is preset in every node during deployment. The counter value C_i used by the key selection protocol for all the sensor nodes is initialized as:

$$\forall_{i=1}^{N} C_i = Seed_i$$

After deployment every node generates its 9 data encryption keys as:

$DK_1 = h(k_i, k_{i-1})$ $DK_2 = h(DK_1, k_i)$
$DK_3 = h(DK_2, DK_1)$ $DK_4 = h(DK_3, DK_2)$
$DK_5 = h(DK_4, DK_3)$ $DK_6 = h(DK_5, DK_4)$
$DK_7 = h(DK_6, DK_5)$ $DK_8 = h(DK_7, DK_6)$
$DK_9 = h(DK_8, DK_7)$

The parameters r_i and r_{i-1} will be discussed later while generating a key for re-keying.

3.5 KEY SELECTION PROTOCOL

Every sensor node maintains a key pool *kp* of 9 keys, which are generated by the node immediately after its deployment. We are limiting the N_{DK} as 9 since our key selection protocol uses a function SoD that always results in a single digit. To increase the security, N_{DK} can be increased but will lead to increase in computation overhead during key generation and re-keying. For each data transmission, the node *i* selects k^{th} key from its key pool as

$$x = \left((ID_i >> \varepsilon_{i,\tau_l}) \times \varepsilon_{i,\tau_l} \right)$$

$$kn_i = \begin{cases} SoD(x) & if\ x > 9 \\ x & if\ x \le 9 \end{cases} \quad (1)$$

where ε_{i,τ_l} is the counter value at τ_l^{th} time interval of i^{th} sensor node which is initialized with the $Seed_i$ and will be incremented as $\varepsilon_{i,\tau_l} = \varepsilon_{i,\tau_l - 1} + 1$ for each constant time interval T_{sec}. τ_l is the time interval which is initialized with 0 and is incremented by T_{sec} for each time interval. That is,

$$\tau_l = \tau_{l-1} + T_{\sec} \quad where\ \tau_0 = 0$$

Now key *k* to encrypt the current message *msg* chosen from *kp* as

$$k = kp\,[kn_i]$$

and the encrypted message E_{msg} is

$$E_{msg_i} = E\left(k, (msg_i, ID_i, kn_i, T_i)\right)$$

T_i is the time stamp at which the i^{th} node transmits a packet and kn_i is the key number in *kp*.

The keys that are generated by all the nodes of a group will be same, but the selection of key for the current communication will not be same. Fig. 2 shows the packet format that carries the data; it includes the type of packet, destination ID, source ID and encrypted message which contain the value, source node ID and the key number *kn* which is used for current encryption.

Pkt_Type	*Dest ID*	*Src ID*	E_{msg}

Figure 2. Data Packet Format

Fig. 3 illustrates how the BS identifies the counter value ε of the nodes *a* and *b* which is used for key selection. Every node in the network will be maintaining a counter value which is initialized with a seed value during the deployment. In our example scenario, counter value of node *a* and node *b* are initialized with $seed_a$ and $seed_b$ respectively. This counter value is incremented by 1 for each constant time interval T_{secs}. The nodes can be deployed at any time interval. During the deployment of sensor nodes, the BS maintain the time interval $\tau_{dep,node\_num}$ at which the nodes are deployed. In this scenario, we assume that node *a* is deployed in 0^{th} time interval τ_0 and node *b* is deployed in 2^{nd} time interval τ_2 of *BS,* so

$$\tau_{dep,a} = 0,$$

$$\tau_{dep,b} = 2$$

Data transmission between node *a* and BS is occurred in different time slots. Data transmission between node b and BS is occurred in same time slot. From this scenario we prove that the key selection protocol chooses the right key when the BS receives the packet at same time slot and different time slots.

Node *a* transfers a packet at time T_a during the time interval τ_3. It computes the key *k* to encrypt the msg_a using the key selection protocol as:

$$\varepsilon_{a,\tau_3} = 3 + Seed_a$$

$$x = SoD\left((ID_a >> (3 + Seed_a)) \times (3 + Seed_a)\right) \quad (2)$$

$$kn_a = x$$
$$k = kp[kn_a]$$

This key k is used to encrypt the msg_a

$$E_{msg_a} = E\left(k, (msg_a, ID_a, kn_a, T_i)\right)$$

BS receives this encrypted message E_{msg_a} from node a at the time $T_{s,a}$ during the time interval τ_4. *BS* then selects the key k from its key pool as:

$$\alpha = T_{s,a} - \delta_{d,a} - \delta_{diff}$$

$$T_{s,a} = T_a + \delta_{d,a} + \delta_{diff}$$

$$\therefore \alpha = T_a \quad (3)$$

$$\beta = T_{s,a}$$

$$x = SoD\left((ID_a >> \varepsilon_{a,\tau_3}) \times \varepsilon_{a,\tau_3}\right)$$

$$\gamma = \tau_4 \times T_{\text{sec}} = 4T_{\text{sec}}$$
$$\varepsilon_{Ba,4} = 4 + Seed_a - \tau_{dep,a}$$
$$= 4 + Seed_a - 0$$

$$\varepsilon_{Ba,4} = 4 + Seed_a \quad (4)$$

The time at which the packet arrived is in 3^{rd} time interval and $T_{s,a}$ is in 4^{th} time interval so $\alpha > \gamma$ && $\beta < \gamma$

$$y = SoD\left((ID_a >> \varepsilon_{B_a,4} -) \times (\varepsilon_{B_a,4} -1)\right)$$

By using equation 3 and 4,

$$y = SoD\left((ID_a >> 4 + (Seed_a -1)) \times (4 + Seed_a -1)\right)$$

$$y = SoD\left((ID_a >> (3 + Seed_a)) \times (3 + Seed_a)\right) \quad (5)$$

$$kn_B = y$$

$$k = kp_a[kn_B]$$

BS uses this key k, to decrypt the E_{msg_a}

$$Payload = D(k, E_{msg_a})$$
$$= (msg_a, ID_a, kn_a, T_i)$$

From the equation 2 and 5 it is proved that kn_B and kn_a are same. So BS states that the packet is not an attacked packet.

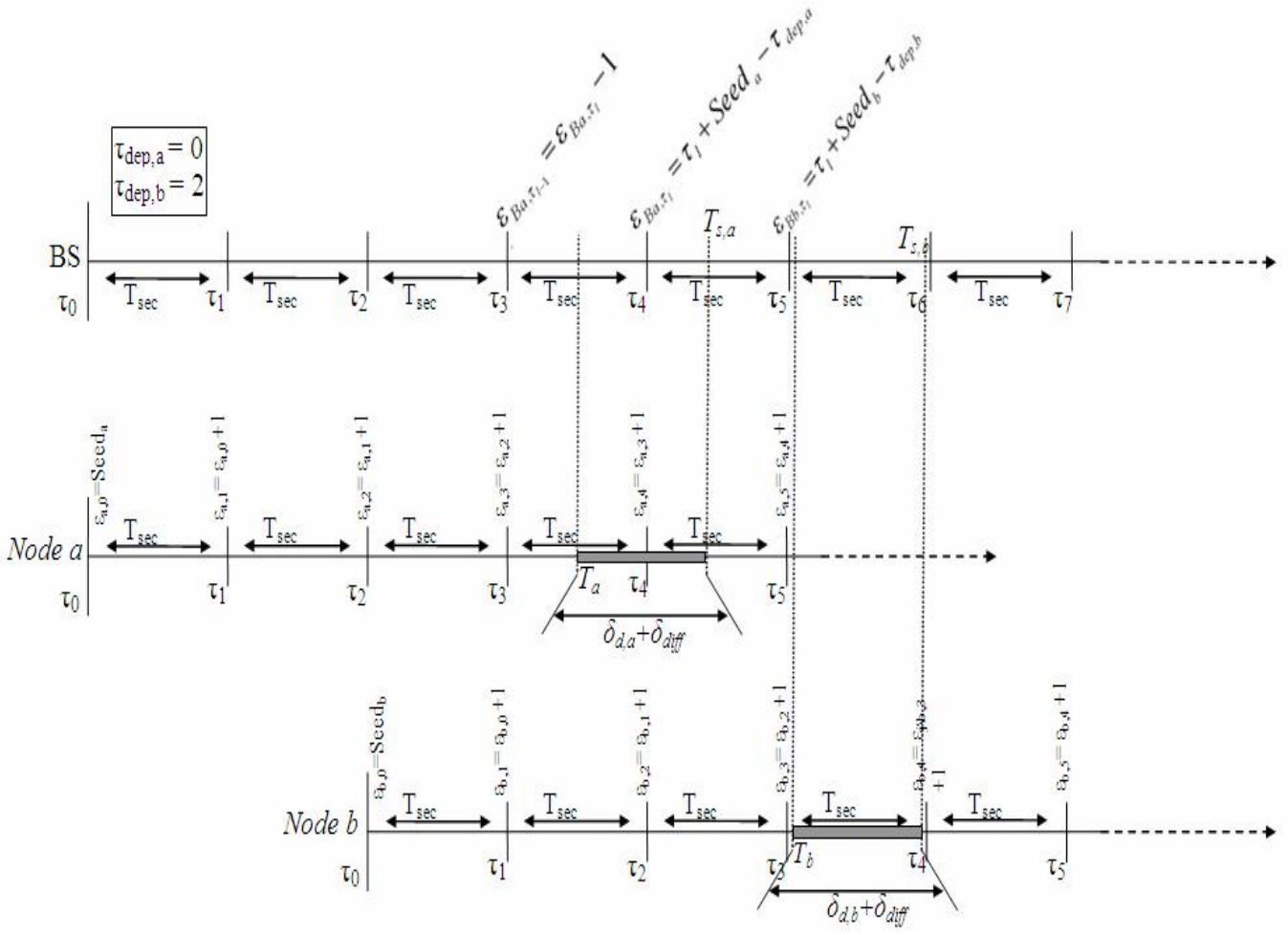

Figure 3. An illustration of increase in counter value among the sensor nodes a and b

3.6 RE-KEYING

In our proposed approach, re-keying is initiated by the sensor node only if any two (other than the last two) consecutive keys are invalidated (compromised). Once all the sensor nodes are ready to deploy in the field, two parameters r_i and r_{i-1} have to be preset. A new re-keying key will be generated by one-way hash function to communicate with the Base Station. This (r_i, r_{i-1}) pair is different for every nodes. Like the Data Encryption keys, the consecutive re-keying keys are also be generated using the previous keys. The protocol for Re-keying mechanism between a node and BS is given below:

Step 1. (Node → BS)

First the node that needs to re-key the existing data encryption keys will send a request to the BS using RKRQ message. This message includes the node ID, its existing group number and the hash value generated by one way hash function. This information is included in this message by encrypting the same using the re-keying encryption key.

$$RKRQ, BS, SrcID, ERKn_i (SrcID, GrpNo, H (SK, SrcID \oplus GrpNo))$$

Step 2. (BS → Node)

Next the BS has to authenticate the node using RKA1 message before sending the parameters for re-keying the data encryption keys. This hash value is generated by using SK and RDT. This information is included in this message by encrypting the same using the re-keying encryption key. All the upcoming messages regarding the re-keying operation uses the re-keying keys itself.

$$RKA1, DestNode, BS, ERKn_i (DestID, H (SK, RDT), RDT, T1)$$

Step 3. (Node → BS)

After receiving the message RKA1, the sensor node generates a hash value using RDT and T1 and compares with the hash value sent by the BS. Then the sensor node authenticates with BS using RKA2 message. This message includes the hash value and the timestamp. The message used for generating hash value is the XOR value of the random text, the time stamp T1.

$$RKA2, BS, SrcNode, ERKn_i (H (SK, RDT \oplus T1), T2)$$

Step 4. (BS → Node)

Now the BS compares the hash value in the RKA2 message with hash value generated by itself using the RDT, T1 and T2. If both are same, the BS sends the parameters for generating the data encryption key to the sensor node using the RKPM message. This message includes the node ID, the parameters, another RDT, timestamp T3 and the hash value generated by SK and $k_i \oplus k_{i-1} \oplus T2$.

$$RKPM, DestNode, BS, ERKn_i (DestID, k_i, k_{i-1}, H (SK, k_i \oplus k_{i-1} \oplus T2), RDT, T3)$$

Step 5. (Node → BS)

Finally the node that receives the parameters has to send an acknowledgment to BS using RKPA message. This message includes the timestamp and the hash value of RDT ⊕ T3.

$$RKPA, BS, SrcNode, ERKn_i (H (SK, RDT \oplus T3), T4)$$

The re-keying protocol requires five transactions in order to complete the process for a single node. If there are N numbers of nodes in the network, it consists of N/N_G nodes per group. So the number of communications N_C in the network during re-keying is,

$$N_C = (N/N_G) * 5$$

Since re-keying occurs occasionally, it does not increase much communication overhead to the network.

3.7 CONNECTIVITY

The grouping in the network does not mean that, the nodes have to communicate only through the nodes that belong to the same group. We introduced grouping in order to maintain only different sets of keys by the groups. Any nodes can send the data to the sink via the intermediate nodes that belongs to any group. So it is not mandatory to know the key for the current communication by the intermediate nodes that forward the packet to sink. Hence the connectivity is not considered to be an issue in our security scheme.

3.8 TIME SYNCHRONIZATION

Since the key management in our method requires time to be synchronized between the nodes in order to maintain the correct counter values in BS as well as in the sensor nodes, we are using Gradient Time Synchronization Protocol (GTSP) [29] which synchronizes the clock accurately in decentralized fashion. Using GTSP, the node synchronizes its logical clock by exchanging beacons for every 30 seconds which consists of the timing information such as current logical time and relative logical clock rate with its neighbors. After receiving the beacons from the neighbors, the node update its absolute clock rate and its logical clock offset. Every node maintains a neighbor table which consists of logical clock value, the relative logical clock rate and last beacon arrival timestamp of their neighbors. This protocol is robust against link and node failures. This protocol requires each node to broadcast only the time information during the synchronization period, the communication overhead is minimum. But the GTSP is vulnerable to time synchronization attacks. Any malicious node can send false synchronization messages to the neighboring nodes and claim to be legitimate. To provide security for GTSP, filters [30] have been added into the architecture of GTSP as shown in Fig. 4. We have used the filters such as logical clock rate filter; logical time filter and timestamp filter. The frequency of sending beacons by a node is set to 30 seconds which is increased after the synchronization period. Firstly, when a beacon is received from its neighbor, the node checks the timestamp of the last received beacon. If the time difference is less than 30 seconds, it adds the sender node ID in the timestamp blacklist filter.

Secondly, the node computes the logical clock rate. If the difference between the received logical clock rate and the most recent logical clock rate of the neighbor is more than the accepted value, it adds the sender node ID in the logical clock rate filter. Finally, the node verifies the received logical time of its neighbor with its own logical time. If it is less than the current logical, then it adds the neighbor node ID in the logical

blacklist filter. Based on these three filters, the legitimate nodes can filter the beacons sent by the nodes in the blacklist filters.

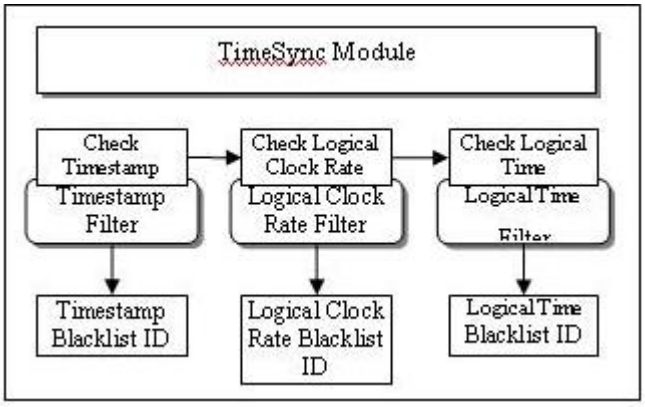

Figure 4. Modified TimeSync Module with three filters

4. Storage Requirement

Storage requirement of our proposed security scheme falls into two categories.

1) Storage at each node.
2) Storage at the BS.

Let L_n - bit length of the Node identifier
L_{sk} - bit length of the SK at each node
L_k - bit length of the keying parameter K_i and K_{i-1}
L_r – bit length of the re-keying parameter r_i and r_{i-1}
L_c – bit length of the counter
L_{dk} – bit length of the generated keys
L_{skpalg} – bit length of Skipjack algorithm

Storage at node in our approach is given as

$$SR_n = L_n + L_{sk} + 2L_k + 2L_r + L_c + 9\,L_{dk} + L_{skpalg}$$

Storage at BS for *m* number of nodes in our proposed scheme is quantized as:

$$SR_{bs} = \sum_{i=1}^{m}(L_{n_i} + \tau_{dep,i} + \varepsilon_{Bi,\tau_l}) + L_{sk} + N_G \cdot (2L_k + 2L_r) + L_{skpalg}$$

Considering 80-bit keys uniform, node is 32 bit long, counter is 8 bit long, the keying and re-keying parameters are 320 bits long and the storage requirement for skipjack algorithm under CBC mode is 21366-bits. So, in our proposed scheme the total storage needed at node SR_n = 6.75KB and the storage needed at base station by considering the number of groups as 2, SR_{bs} = 7.32KB. Fig. 5 shows the memory consumption (both code and data memory) at nodes and base station.

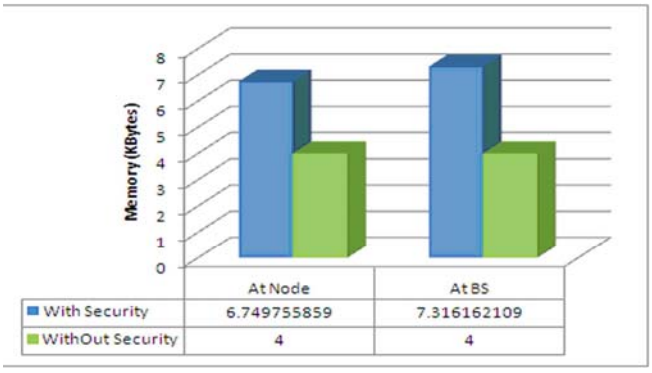

Figure 5. Memory Consumption at nodes and base station

5. Security Analysis

In this section we explain how our approach detects various attacks such as Packet Replay attack, Sybil attack and DoS attack.

Replay Attack: Replay attack occurs when an attacker captures the packet at some point of time and then replays the same at later point of time without any modification.

Sybil Attack: In Sybil attack, any particular node illegitimately claims for several identities [22][23]. The Sybil node act as original node and can introduce false packets into the network and disrupt the purpose of the network.

DoS Attack: In DoS Attack, the attacker captures the key processing request pattern and raises these requests frequently and blocks the service availability to others.

When the base station receives the packet from the i^{th} node, it identifies the key number kn_B using (1) to decrypt the message E_{msg}. If the BS receives the packet from node *i* in the time that falls in the same time interval τ_l at which the node *i* sends the packet, then the BS uses the counter value ε_{Bi,τ_l}; otherwise if the packet reaches the BS in the next time interval τ_{l+1}, it uses the previous counter value $\varepsilon_{Bi,\tau_l}-1$. The key *k* is selected using kn_B and the E_{msg_i} is decrypted by k to obtain the payload.

If the key *k* cannot decrypt the received encrypted packet, it will be treated as an illegal packet. Then the base station tries to decrypt the received encrypted packet using the remaining valid keys. If the packet cannot be decrypted by any of the remaining valid keys, then the BS identifies the packet has been corrupted. If any one of the remaining valid keys decrypts the packet, then the BS verifies the timestamp T_i. If the packet is not a fresh packet, then the BS declares that this is a replay packet.

If the packet is a fresh one, then the BS declares that this is a Sybil attack and it broadcast a message to invalidate the key number kn_i of the group where the Sybil node exists and will send a command to that node not to send any data for a configurable period of time.

In our proposed approach, the choice for DoS attack is the re-keying request packet. An attacker can frequently send re-keying request and launch the DoS attack. In our approach, re-keying request comes from the node only when any two consecutive keys are invalidated or the lifetime of the keys have been expired. Base stations will maintain this information for each node. So, if the rate of re-keying requests is coming frequently, then base station can conclude for possible DoS attack and drop the packets from that node. Base station can also send a broadcast packet to stop processing request from the attacking node for an interval

6. Comparison of our Approach with other Security Solutions

Comparison of SNEP and LEDS with our approach is given in Table 2. We provided comparison from the perspective of memory requirement, communication and computation overhead and some other basic security parameters such as data integrity, confidentiality, availability, authentication, etc. Our approach provides re-keying, but LEDS regenerate keys only if the nodes are dislocated [20]. SNEP provides security, but

attack model is not discussed in that paper. We provide authentication support only during re-keying process. We provided the computation overhead in terms of number of rounds required by one-way hash function to generate the keys and skipjack algorithm for encryption/decryption process.

Table II.
Comparison of our Approach with other Security Solutions

Parameters		**SPINS – SNEP** [18]	**LEDS** [20]	**Our Approach**
Memory Requirement with respect to storage of keys		3 [24]	26 if T=5	9
Cryptography Mechanism used		Symmetric	Symmetric	Symmetric
Data Integrity support		Yes [24]	No	Yes
Confidentiality support		Yes [24]	Yes	Yes
Availability support		No [24]	Yes	Yes
Re-keying Support		Yes	Only during the node dislocation.	Yes
Attack Identification		Replay attack	Alternation, False-data injection attack	Replay, DoS, and Sybil attacks
Authentication Support		Yes [24]	Yes	Only during Re-keying
Communication Overhead During data transmission		8 Bytes	1/4th of original Message Length. 9 Bytes if Message length is 36 Bytes.	8 Bytes
Computation Overhead	Key generation	Initially 3 keys. 3× (0 – 255) rounds Then one key for every session	26 times Pseudorandom Function is called.	9 × 10 Rounds (During Bootstrapping)
	Re-keying	0 – 255 rounds to calculate one Key	26 times Pseudorandom Function is called.	9 × 10 Rounds
	Encryption/ Decryption	0 – 255 Rounds	Assuming RC5 0-255 Rounds	32 Rounds

In SNEP, they have used the MAC (CBC-MAC) function to generate the key. CBC-MAC uses CTR-RC5 block cipher algorithm to generate MAC. RC5 algorithm takes 0-255 rounds to produce the cipher text.

7. Performance Evaluation

We use Castalia [25][26] to evaluate the performance of our approach. In our simulation study, we use 100 nodes with manual deployment in 100 × 100 m and initial energy is set to 50 J. We measure the performance using the following metrics:

- *Packet Delivery Ratio:* The total number of packets received is divided by the total number of packets sent from the source.
- *Network Availability:* Availability can be measured by means of the lifetime of the secure wireless sensor network under various conditions.

Fig. 6 shows that our mechanism maintains good packet delivery ratio of 70 % even the number of malicious nodes increases up to 50. The increase in number of nodes will proportionally increase the number of false packets over the network. In our security mechanism, the BS identifies the false messages and broadcast the command not to forward the false message from the malicious nodes. The node that receives this command will simply discards the false messages, so that the normal traffic in the network is maintained which provides good delivery ratio.

In SNEP, all the security threats are identified and discarded at the BS. So network is affected with high congestion because of the false messages transmission. So SNEP drops more legitimate packets when the malicious nodes increased. But LEDS reduces the false message transmission and the valid packet dropping by using the en-route-filtering operations so it achieves the best delivery ratio than all other approaches.

For our DoS attack simulation, we keep a constant attack rate of packets and calculate the average energy at the nodes for over a period of time. Fig. 7 shows the effect of DoS attack on network energy. SNEP does not identify the DoS attack. So it consumes greater energy due to DoS attack. In our approach, we stop processing the packets at nodes for some time interval, whenever the DoS attack is detected. In LEDS, DoS attack is prevented by using en route filtering which needs extra energy than our approach.

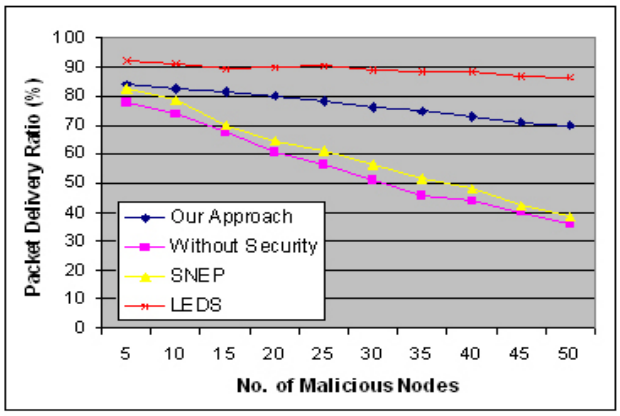

Figure 6. Effect of Packet Delivery Ratio on increase in Malicious Nodes

Fig. 8 shows the effect of replay attack on network energy when five attackers replay the packets at the rate of 10 to 50 copies per second. SNEP uses implicit counter maintained at both ends to protect against replay attack. In SNEP, if the replay packet is identified by the destination, it simply discards the replay packet and the remaining replay packets are still forwarded by the nodes to the destination. But in our mechanism, if the replay attack is identified, the BS sends a command not to forward the packets from the node where the replay attack is launched for a configurable period of time. This will reduce the energy consumption during that time period.

Because of the communication overhead to detect the replay attack, SNEP consumes greater energy than the network without security mechanism. LEDS effectively controls the reply attack but because of its high communication overhead it consumes greater energy than our approach.

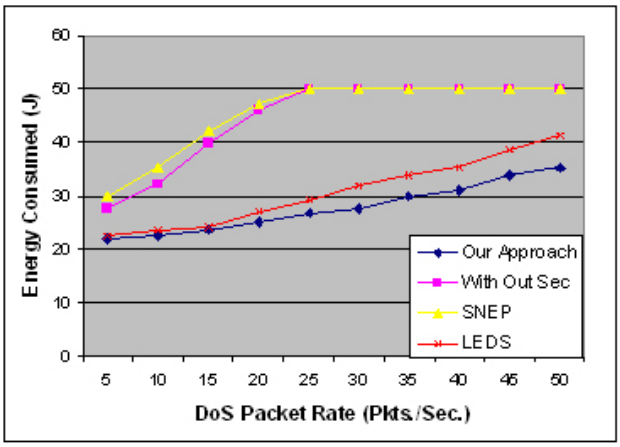

Figure 7. Effect of DoS attack on network Energy

Network availability [27][28] can be improved by means of increasing the lifetime of the secure wireless sensor network under various conditions. Fig. 9 shows the percentage of Network Availability over the time.

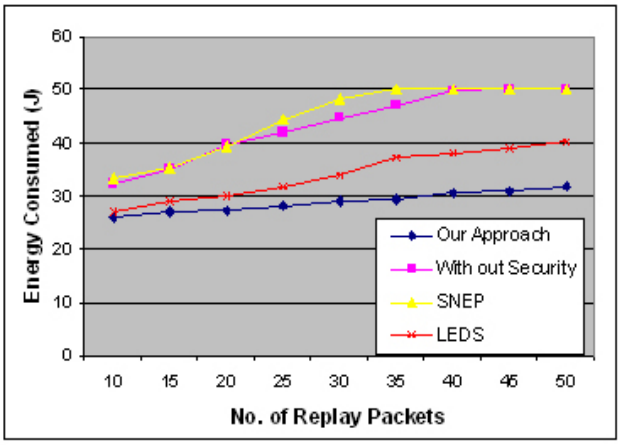

Figure 8. Effect of Replay Packets on network Energy

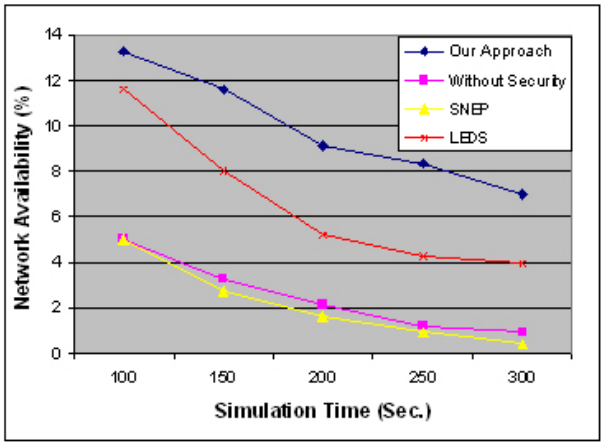

Figure 9. Percentage of Network Availability over the time

Because of the computation and communication overhead, SNEP drains the nodes faster than without security mechanism. LEDS and our approach effectively control the security threats so node availability in both the methods is maintained for longer period. But in our approach, computation and communication cost is much lesser than LEDS, we achieved the higher network availability than LEDS.

8. Conclusion and Enhancements

Our approach uses one-way hash function to dynamically generate the keys that avoid transmission of key during runtime. In order to minimize the memory overhead, we have introduced grouping among nodes in the network that maintains different sets of keys. Our approach identifies the attacks such as Replay attack, Sybil attack and DoS attack. We present our mechanism by analyzing the parameters such as network availability, packet delivery and network energy on replay and DoS attacks. In our proposed mechanism, scalability can be still increased by introducing the Clustering concept in order to reduce the traffic and overhead to the Base Station

References

[1] A. A. Seema, K. D. Kulat, R. V. Kshirsagar, "WSN based Low Cost and Low Power EPM Design and Field Micro-Climate Analysis using Recent Embedded Controllers", International Journal of Computer Applications, Vol. 12 (6), 12 – 22, December 2010.

[2] M. Hefeeda and M. Bagheri, "Forest Fire Modeling and Early Detection using Wireless Sensor Networks", Ad Hoc & Sensor Wireless Networks, Vol. 7, 169–224, 2009.

[3] I. Shin, Y. Shen, Y. Xuan, My T. Thai and T. Znati, "A novel approach against reactive jamming attack", Ad hoc & Sensor Wireless Networks, Vol. 0, 1-25, 2010.

[4] X. Chen, K. Makki, K. Yen, N. Pissinou, "Sensor network security: a survey", IEEE Communications Surveys & Tutorials, Vol. 11(2), 52-73, 2009.

[5] R. Shivangi, P. Amar, P. Kishore Babu, S. Prateek, S. Ashish and S. Shveta, "Wireless sensor networks: A Survey of Intrusions and their Explored Remedies", International Journal of Engineering Science and Technology, Vol. 2(5), 962-969, 2010.

[6] X. Du and H. H.Chen, "Security in Wireless Sensor Networks", IEEE Wireless Communications Magazine, Vol. 15, Issue 4, 60-66, 2008.

[7] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, M. Galloway, "A survey of key management schemes in wireless sensor networks", Computer Communications, Vol. 30, 2314–2341, 2007.

[8] R. Di Pietro, L. V. Mancini and S. Jajodia, "Providing secrecy in key management protocols for large wireless sensors networks", Journal of AdHoc Networks, Vol. 1(4), 455-468, 2003.

[9] C. L. Chen and C. T. Li, "Dynamic Session-Key Generation for Wireless Sensor Networks", EURASIP Journal on Wireless Communications and Networking, Vol. 2008.

[10] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", 9th ACM conference on Computer and communications security, Washington, DC, USA. 41–47, November 2002.

[11] H. Chan, A. Perrig, and D. Song, "Key Distribution Techniques for Sensor Networks, "Wireless Sensor Networks, ch.1, pp. 277-303,2004.

[12] H. Chan, V. D. Gligor, A. Perrig, and G. Muralidharan, "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 3, pp. 233-247, Jul.-Sep. 2005.

[13] Y. Cheng, and D. P. Agrawal, "Efficient Pairwise Key Establishment and Management in Static Wireless Sensor Networks," in Proc. Mobile Ad hoc and Sensor System Conference (MASS), pp. 550,Washington, DC, USA, Nov. 2005.

[14] A. Parakh and S. Kak, "Online data storage using implicit security, "Information Science, vol. 179, no. 19, pp.3323-3331, Sep. 2009.

[15] E. K. Wang, Y. Ye, "An Efficient and Secure Key Establishment Scheme for Wireless Sensor Network", 3rd International Symposium on Intelligent Information Technology and Security Informatics (IITSI), 511 – 516, 2010.

[16] T. Li, H. Wu, X. Wang and F. Bao, "SenSec: Sensor Security Framework for TinyOS", Second International Workshop on Networked Sensing Systems, San Diego, USA. 145-150, 2005.

[17] K. Sharma, M. K. Ghose, "Complete Security Framework for Wireless Sensor Networks", International Journal of Computer Science and Information Security (IJCSIS), Vol. 3(1), 2009.

[18] A. Perrig, R. Szewczyk, V. Wen, D. Culler and T. D. Tygar, "SPINS: Security protocols for sensor networks", 7th Annual International Conference on Mobile Computing and Networking (MobiCom 01), ACM Press, 189-199, 2001.

[19] C. Karlof, N. Sastry and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks", 2nd International Conference on Embedded Networked Sensor Systems, ACM Press, 162-175, 2004.

[20] K. Ren, W. Lou and Y. Zhang, "LEDS: Providing location-aware end-to-end data security in wireless sensor networks", IEEE Transaction on Mobile Computing, Vol. 7 (5), 585– 598, 2008.

[21] Y. W. Law, J. Doumen and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks", ACM Transactions on Sensor Networks, Vol. 2(1), 65–93, 2006.

[22] J. Newsome, E. Shi, D. X. Song and A. Perrig, "The Sybil attack in sensor networks: analysis & defenses", IPSN'04, 259-268, April 2004.

[23] Q. Zhang, P. Wang, D. S. Reeves and P. Ning, "Defending against Sybil attacks in sensor networks", ICDCS Workshops, 185-191, 2005.

[24] R. Shaikh, S. Lee, M. Khan and Y. Song, "LSec: Lightweight security protocol for distributed wireless sensor networks", 11th IFIP International Conference on Personal Wireless Communications (PWC'06), 4217 of LNCS, 367—377, September 2006.

[25] A. Boulis, "Castalia, a simulator for wireless sensor networks and body area networks", version 2.0, user's manual, May 2009.

[26] Castalia: A Simulator for WSN, http://castalia.npc.nicta.com.au/

[27] M. Sharifi, S. Pourroostaei and S. Sedighian, "Improving Availability of Secure Wireless Sensor Networks. 4th International Conference on Sciences of Electronic, Technologies of Information and Telecommunications (IEEE SETIT 2007), 1-6, March 2007.

[28] D. Djenouri and L. Khelladi, "A survey of security issues in mobile Ad hoc and Sensor Networks", IEEE communications surveys & tutorials, fourth quarter, Vol. 7(4), 2005.

[29] P. Sommer, R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks", Proceedings of the 2009 International Conference on Information Processing in Sensor Networks, p.37-48, April 13-16, 2009.

[30] D. Huang, K. You, and W. Teng, "Secured Flooding Time Synchronization Protocol", in Proc. MASS, 2011, pp.620-625.