SeFER: Secure, Flexible and Efficient Routing Protocol for Distributed Sensor Networks

Cagil Can Oniz, Sinan Emre Tasci, Erkay Savas, Ozgur Ercetin and Albert Levi
Sabanci University, Faculty of Engineering and Natural Sciences,
Orhanli, Tuzla, 34956 Istanbul, Turkey
{cagilo,sinanemre}@su.sabanciuniv.edu
{erkays,oercetin,levi}@sabanciuniv.edu

Abstract – **In this paper, we present a secure, flexible, and efficient routing protocol for sensor networks based on random key pre-distribution. Random key pre-distribution provides an easy way to manage the keys in a large-scale network without using public key cryptography, which is considered to be expensive. Our protocol aims to establish secure paths in a sensor network between a controller and a set of nodes where each node has been assigned a set of randomly chosen keys out of a key pool. A common model for sensor networks assumes a tree of sensor nodes delivering information to the controller according to an inquiry sent into the network. However, if we require the communication to be secure among the sensor nodes, such a tree cannot always be built efficiently. For example, when the nodes are assigned randomly chosen keys, many of them may not communicate directly since they do not often share a common key. However, these two nodes may communicate indirectly but securely over a multiple hop path where each pair of nodes on this path shares a common key. Our protocol bridges the gap between these two cases by providing the methods for nodes to securely share their keys and communicate directly so that the efficiency of communications is increased without jeopardizing the security. In this way, our protocol generates secure and efficient routes. We also provide simulation results for our protocol demonstrating that, for a small number of keys stored at each node, the average path length is smaller. However, the gains due to our protocol diminish as the number of available keys at each node increases since two nodes within communication range of each other are more likely to have a key in common.**

Index Terms – **sensor networks, secure routing, random key distribution, nonce chains**

I. Introduction and Background

Sensor nodes are tiny wireless communication devices that have limited energy, computational power and memory resources. A collection of many sensor nodes which gather information from the environment and send it to a controller node is called Distributed Sensor Network (DSN). A controller node in general is mobile and more powerful than a sensor node. The controller node accumulates and interprets the data received from sensor nodes. Sensor nodes usually have limited transmission and observation range and can cover a large physical area only by networking a large number of sensor nodes. Thus, scalability is a major issue in the protocol design for sensor networks.

Security and secure routing is an important issue in DSNs, especially in applications where data authenticity, confidentiality and/or integrity are required. Achieving security in DSNs is a hard task because of the limited resources of sensor nodes. Thus, applicability of cryptographic protection techniques is also limited. For example, public-key cryptography based key exchange protocols, such as Diffie-Hellman, are not viable in sensor nodes.

Security problems in ad-hoc networks are similar to security problems in sensor networks. Some of the security issues in ad hoc networks have been given in [3]. Although the security problems are similar, the solutions to these problems are quite different due to the differences (i.e., resource limitations of DSNs) between sensor and ad-hoc networks. For instance, some ad hoc security protocols use public-key cryptography [3], [4], [5], [6], [7], [8]. As stated previously, public-key cryptography is not suitable for sensor networks due to resource limitations. There are some security protocols for ad hoc networks which use symmetric cryptography [9], [10], [11]. Perrig et al. [12] presented two security protocols μTESLA and SNEP. The protocol μTESLA is for authenticated broadcast and the protocol SNEP is for the authentication of freshness and confidentiality. Tatebayashi et al. [13] worked on key distribution for resource limited mobile devices. Boyd and Mathuria [14] presented a survey on previous authentication and key distribution methods in mobile environments. In [16], Di Pietro, Mancini and Jajodia proposed a key establishment protocol in which forward and backward security of session keys are provided. In other words, the compromise of a session key does not lead to the compromise of previous or future session keys. Du, Deng, Han and Varshney [17] proposed a key pre-distribution scheme which improves resilience of network. In this protocol, if the number of compromised nodes is less than a certain threshold value, then the probability that the uncompromised nodes will get affected is low. This property increases the cost of a successful attack. In [18], a general framework for key pre-distribution which provides tolerance to node capture in an efficient way is presented by Liu and Ning. Zhu, Setia and Jajodia [19] proposed a key management protocol that provides support for numerous symmetric keying mechanisms such as individual keys, pairwise shared keys, cluster keys and group keys.

Key distribution is the starting point of any security protocol. The easiest way to distribute keys in a large DSN is *key pre-distribution*, in which the necessary keys are stored in sensor nodes before deployment. In this paper, our study is founded on the key pre-distribution method described by Eschenauer and Gligor in [1]. It is based on probabilistic key sharing among the nodes of a random graph. Each node runs a shared key discovery protocol to find the neighbors with whom they share a key. Before deployment, we distribute a key ring of k keys that are randomly selected from a large key pool P. Despite the fact that a pair of nodes may not share any keys, if a path of nodes exists between these nodes, then key exchange may be performed through that path. Therefore, each node does not have to be (cryptographically) connected to all other n-1 nodes, which is the case in the pair wise key sharing method.

In this paper, we propose a secure and flexible routing protocol based on the key pre-distribution mechanism mentioned above. Our primary aim is to find routes from each sensor node to the controller with all links secured. The proposed protocol is quite flexible such that the sensor nodes may still establish secure routes even if they lack energy and memory by sacrificing the path length. Another useful feature is that selective revocation of a compromised node is possible.

After each node discovers shared keys with its neighbors as described in [1], our routing protocol starts. There are six phases. In the *Level-One Initialization Phase*, the controller and nodes in the wireless range of the controller mutually authenticate themselves and the controller distributes the session key to be used in further phases. In the *Route Learning Phase*, each node forwards messages containing route information to their downstream nodes and an initial set of routes is established. In the *Authentic Neighbor and Shorter Path Discovery Phase*, nodes broadcast messages in order to discover shorter paths to the controller. If shorter paths are found, these paths must be secured by assigning a key to that path. This key is exchanged in the *Key Exchange Phase*. Should the controller detect that a security breach has occurred during the execution of the routing protocol, it may invalidate the session key by starting the *Session Key Expiration Phase*. If a legitimate sensor node is compromised, the *Revocation Phase* may be started by the controller in order to invalidate the key ring of the compromised node.

The rest of the paper is structured as follows. We state our system assumptions and give some definitions and notations in Section 2. In Section 3, we explain some preliminary information about Eschenauer and Gligor's work [1]. In Section 4, we describe our routing protocol. Some attacks and countermeasures are discussed in Section 5. Simulation results are given in Section 6 and we conclude the paper in Section 7.

II. Assumptions and Notations

In a distributed sensor network, all messages are broadcast. However, for the sake of clarity, we will sometimes refer to sending of a message created specifically for a certain node as unicasting. We assume that for each point-to-point message between the nodes, the integrity of data is protected by appending a MAC (Message Authentication Code) value to the message; again, for the sake of simplicity, we do not show these MAC values in the protocol.

A secure routing protocol must be resilient to replay attacks. In our protocol, we consider using nonces with time stamps. We do not explicitly show these nonce and time stamps in the protocol for the sake of clarity and readability. Our method requires loose synchronization. Each message will contain a nonce value and a time stamp, indicating a message expiration date. When a node receives such a message, it must save the nonce value in its memory until the expiration date of the time stamp. If a message with the same nonce value arrives before the expiration date, this message must be discarded, since it is considered a replay. After the expiration date, the same nonce value can be used with another message but with a different time stamp bearing a larger time value.

The proposed protocol uses one-way nonce chains, which is a concept introduced in [12], in order to verify that the Route Learning, Session Key Expiration, and Revocation phases in the protocol are originally initiated by the controller. A one-way nonce chain is a sequence of related secret nonces. In order to generate a one-way nonce chain of length n, the last nonce of the chain N_{n-1} is first chosen at random. Then, a one-way hash function H is successively applied n-1 times as $N_i = \mathrm{H}(N_{i+1})$. Since, H is a one-way function, computing N_i given N_{i+1} is easy, but computing N_i given N_{i-1} is computationally infeasible. In the presented protocol, we use nonce chains that can only be created by the controller and can be verified by all the nodes in the network. In the paper i^{th} nonce of the nonce chain is denoted N_{chain_i}.

We assume that the first nonce, N_0, of the nonce chain has been distributed before the network deployment and that there is a protocol for learning the current nonce of the nonce chain. The mechanism to distribute the first nonce of the nonce chain may be similar to the *Session Key Expiration Phase* (Section 4.5) or *Revocation Phase* (Section 4.6)[1]. The problem of the distribution of such initial parameters in a secure way has been investigated in the literature as well. In [12], a mechanism is proposed in which the controller unicasts the first nonce of each node. However, this is not an efficient method and therefore has scalability problems. In [15], the problem of distributing initial parameters is addressed by broadcasting in an efficient way. Furthermore, a multi-level key chain scheme in which higher-level key chains are used to authenticate the commitments of lower-level ones is introduced in order to lengthen the lifetime and to provide efficiency.

The notation commonly used to express our protocol is given in Table 1.

[1] These two phases use the same method to securely spread two different types of messages to the network. These messages are the *session key expiration message* and *node revocation message*.

Table 1. Notations

Symbol	Meaning
k	Key ring size
P	Pool of keys
K_{AB}	Symmetric key shared between the nodes A and B
K_S	Session key used during routing
$A \rightarrow B: X$	Node A sends X to Node B
$E_{K_{AB}}$(data)	Symmetric encryption of data with a key shared between Node A and Node B
depth	Number of levels of nodes through which routing messages will be forwarded.
N_{chain_i}	ith nonce of the one way nonce chain
t	Timestamp
N_x	Nonce value generated by Node X

III. Preliminaries

The proposed routing algorithm is based on a random key pre-distribution and a shared key discovery phase presented in [1]. A brief description of these phases is given below.

The *Key Pre-distribution* phase consists of five offline steps. First, a large pool of keys P and their identifiers are generated. Second, for each node, a key ring with k distinct keys is randomly selected from the pool P. The number k is called the key ring size. Some key rings may have keys in common because of the random selection. Third, each selected key ring is loaded into the memory of a sensor. Fourth, identifiers of keys in the key ring of the sensor node and sensor identifier are stored on a trusted controller node. Lastly, each sensor node shares a separate key with the controller node. This key is computed by each node and the controller automatically uses all of the keys in the corresponding key ring.

In the *Shared Key Discovery* phase, each node broadcasts key identifiers of its key ring. Doing so, each node finds its neighbors with whom it shares a key.

IV. Routing Protocol Description

In this section, we describe the proposed routing protocol. First, keys are pre-distributed to the sensor nodes and shared keys are discovered by the methods discussed in [1] and briefly overviewed in Section 3.

The controller can communicate with the rest of the nodes indirectly via the *level-one* nodes, which are defined as the nodes in the wireless range of the controller. Forming a route consists of four phases: (1) *Level-One Initialization Phase*, (2) *Route Learning Phase*, (3) *Authentic Neighbor and Shorter Path Discovery Phase,* and (4) *Key Distribution Phase*. After forming the route, the controller may decide to invalidate a session key or to revoke some nodes. For these purposes, the *Session Key Expiration Phase* or *Revocation Phase* may be run respectively. Figure 1 depicts the flowchart of the protocol.

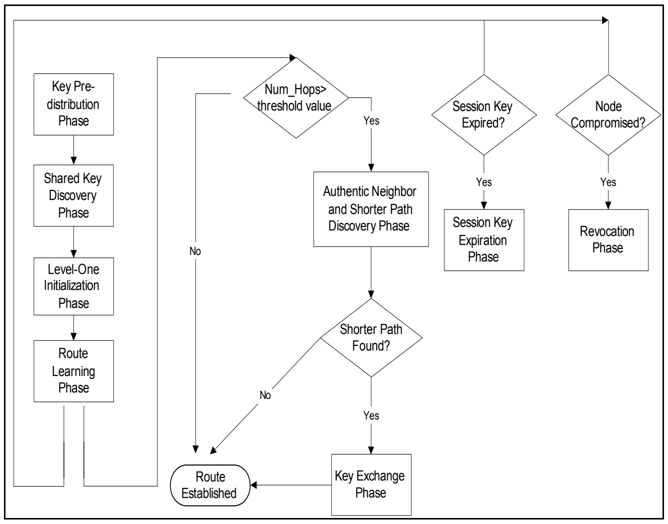

Key Pre-distribution Phase
Shared Key Discovery Phase
Level-One Initialization Phase
Route Learning Phase
Num_Hops> threshold value
Yes
No
Authentic Neighbor and Shorter Path Discovery Phase
Shorter Path Found?
No
Yes
Key Exchange Phase
Route Established
Session Key Expired?
Yes
Session Key Expiration Phase
Node Compromised?
Yes
Revocation Phase

Fig. 1. SeFER Flowchart

A. Level-One Initialization Phase

In this phase, the controller discovers the identities of the level-one nodes. The controller and level-one nodes mutually authenticate themselves. Furthermore, the controller sends a message containing a session key for broadcast authentication, the depth (hop count) of the network, and next nonce of the nonce chain. Figure 2 shows the steps involved in the level-one initialization phase for two level-one nodes A and B.

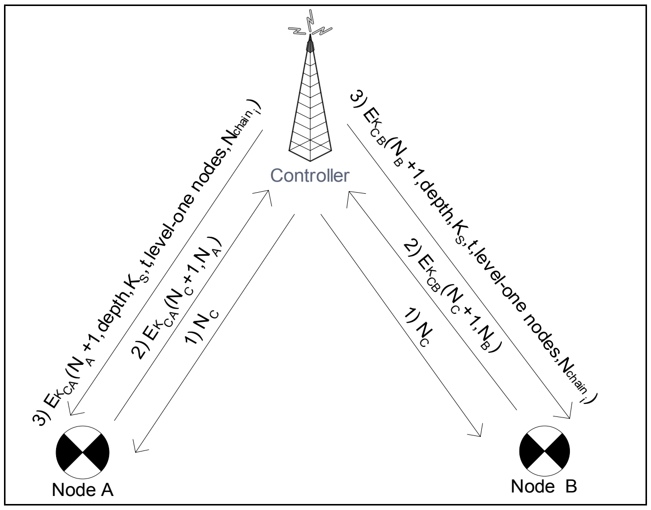

Controller
3) $E_{K_{CA}}(N_A+1,depth,K_S,t,\text{level-one nodes},N_{chain_i})$
2) $E_{K_{CA}}(N_C+1,N_A)$
1) N_C
1) N_C
2) $E_{K_{CB}}(N_C+1,N_B)$
3) $E_{K_{CB}}(N_B+1,depth,K_S,t,\text{level-one nodes},N_{chain_i})$
Node A
Node B

Fig. 2. Level-One Initialization Phase

In the first message, the controller broadcasts a clear-text message N_C, a nonce value produced by the controller. In the second message, the level-one nodes reply by sending N_C+1 encrypted with the key shared with the controller; which is K_{CA} for Node A and K_{CB} for Node B. Note that each node shares a different key with the controller. The second message authenticates Node A and Node B to the controller. The second field in the second message is a nonce value (N_A for Node A, N_B for Node B) produced by each level-one node. This value is used to authenticate the controller in the same way that level-one nodes are authenticated to the controller. In the third message, the controller authenticates itself to the level-one nodes by sending N_A+1 to Node *A* and N_B+1 to Node *B* encrypted with K_{CA} and K_{CB} respectively. The second field in the third message is depth, which is the number of levels of nodes through which routing messages are forwarded. Each node forwarding a routing packet decrements the value of this field by one. The third field in this encrypted message is a session key, K_S, produced by the controller. This session key is used to perform an authentic broadcast and to execute some other operations explained later in Authentic Neighbor and Shorter Path Discovery phase. The session key is not used for any critical operations; neither key exchange, nor data encryption is performed with the session key. The fourth field is a timestamp, which is the expiration time of the session key. The compromise of a session key only gives a malicious node the opportunity of time-limited attacks, such as performing a Denial of Service (DoS) attack until the expiration of the session key (see [2] for detailed information about DoS attacks). Expiration time must be chosen as short as possible but long enough to complete the route setup. A session key may be invalidated before the expiration time by the controller by running the Session Key Expiration phase, which is explained later. The fifth field is the list of level-one nodes. In order to prevent level-one nodes sending routing messages to each other, the controller informs all level-one nodes of the identity of each other. Sending these lists prevents the flooding and receiving of multiple routing messages among level-one nodes. Finally, the last field of the message is the next nonce in the nonce chain, N_{chain_i}. This value can only be created by the controller and used to find evidence when the routing messages are not initiated by a legitimate controller node, as detailed in Section 5.3.

B. Route Learning Phase

The aim of level-one nodes in the *Route Learning Phase* is to spread a route message that includes the session key K_S, timestamp *t* and the next nonce of the nonce chain N_{chain_i} to a number of nodes in the network. The operation starts with the level-one nodes and continues down in the network. At each hop, nodes add their identity to the route message. In this way, an initial set of routes is established. Figure 3 shows the steps involved in the *Route Learning Phase.*

First, a sensor node receives a routing message from one of its secure paths. This message is encrypted with the key shared between the sender of the message, *Previous_Node* and the receiver, *Current_Node*. *Current_Node* decrypts this message and checks the validity of the nonce N_{chain_i}, where N_{chain_i} is the current unpublished nonce in the nonce chain. Second, if the depth field is not zero, *Current_Node* adds its identity to the *Route* field, decrements the *depth* field and forwards this message to its secure paths. These messages are encrypted with the key shared between *Current_Node* and the receivers of this message *Node*$_j$, where *j=1…Number_of_Secure_Paths*.

```
Previous_Node(P)➔Current_Node(C):
                    $E_{KPC}$(depth, $K_S$, t, route, $N_{chain_i}$)

  if($H(N_{chain_i})==N_{chain_{i-1}}$)AND($N_{chain_i}$ not used before)
    if (depth > 0)
       route := route || Current_Node_Identifier
       depth := depth-1
       send_to_Secure_Paths(
         depth,$K_S$, t,route, $N_{chain_i}$
       )
    end
  end
```

Fig. 3. Route Learning Phase

Figure 4 shows an example of the *Route Learning Phase*. The dotted edges in the graph are secure links. Nodes *A* and *B* are level-one nodes. Each node forwards the routing message by decrementing the *depth* field and adding its identity to the *route* field. Furthermore, each message is encrypted with the key shared between the sender and the receiver of the message.

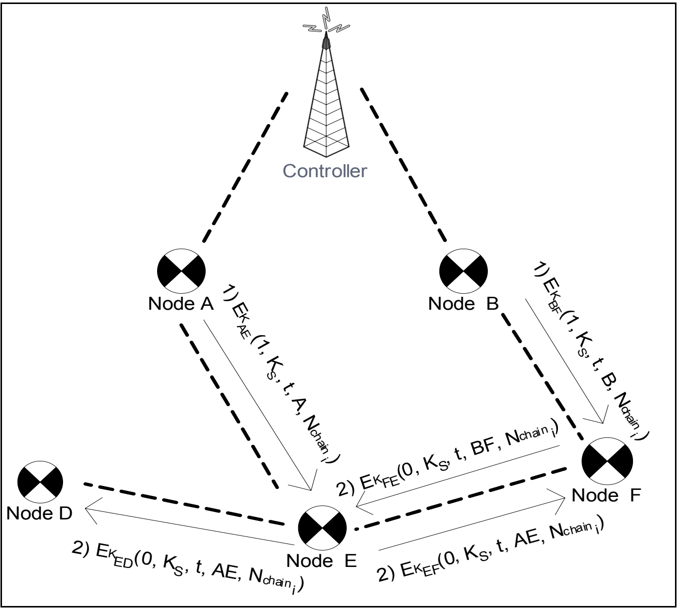

Controller
Node A
Node B
1) $E_{K_{AE}}(1, K_S, t, A, N_{chain_i})$
1) $E_{K_{BF}}(1, K_S, t, B, N_{chain_i})$
2) $E_{K_{FE}}(0, K_S, t, BF, N_{chain_i})$
Node F
Node D
Node E
2) $E_{K_{ED}}(0, K_S, t, AE, N_{chain_i})$
2) $E_{K_{EF}}(0, K_S, t, AE, N_{chain_i})$

Fig. 4. An Example of the Route Learning Phase

C. Authentic Neighbor and Shorter Path Discovery Phase

After the Route Learning Phase, each node receiving the routing message has a route to the controller. However, there may be a shorter but non-secure route to the controller. Such a route implies communication connectivity, but some links are not cryptographically connected since the end-points of these links do not share a key. In the Route Learning phase, only secure routes are established. The Authentic Neighbor and Shorter Path Discovery Phase finds the above-mentioned non-secure, but shorter paths in an authentic way. After finding such a shorter path, we run the next phase, i.e., the Key

Exchange Phase, to remove the cryptographic disconnectedness and secure all of the links in that path.

Figure 5 shows an example of the steps involved in the Authentic Neighbor and Shorter Path Discovery Phase for Node *X*. The dotted lines are secure links. This phase is performed if the length of the path found is greater than a certain threshold value. This operation limits the number authentic neighbor and shorter path phase messages. In Figure 5, Node *X* queries its neighbors to find out if any of them has a shorter path to the controller. First, Node *X* broadcasts a route request message encrypted with the session key K_S thus authenticating the message. The first field in this message is the identity of the node requesting route information (Node *X* in this case). The second field is a nonce value, N_X, created by Node *X*. The third field is the time stamp, *t*, of the nonce value. This field is used to prevent replay attacks. Second, each neighbor of Node *X* responds by a message encrypted with session key K_S. The first field in this response message is the identifier of the responding neighbor. The second field is the N_X+1. If this field is valid, this message is a valid response to Node *X*'s broadcast. The last field is the route of the neighbor to the controller. Node *X* receives multiple responses from its neighbors. Node *X* can adopt the shortest path to the controller among these responses from its neighbors if it is shorter than its current path.

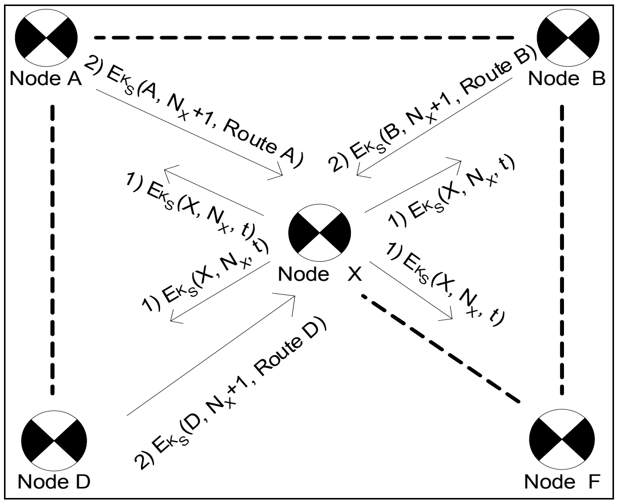


Fig. 5. An Example of Authentic Neighbor and Shorter Path Discovery Phase

D. *Key Exchange Phase*

In the previous phase, Authentic Neighbor and Shorter Path Discovery Phase, nodes have asked their neighbors for a shorter path. If a shorter path exists, this path must be secured. Thus, a one-time key exchange must be completed between these nodes, say, *X* and *Y*. Key exchange is performed through the controller. First, one of the nodes, say *X*, picks a key K_{XY} and encrypts it using the key shared between *X* and the controller. Then *X* sends out the encrypted key to the controller over the existing route between them. The controller decrypts K_{XY} and then re-encrypts it using the key shared between the controller and *Y*. After that, the controller sends out the encrypted key to *Y*. Node *Y* decrypts K_{XY} and sends *X* an acknowledgment message, which is encrypted by K_{XY}. This acknowledges the receipt of the session key by *Y*.

Consider the example in Figure 6. In this figure, lines indicate secure links and Node *A* is the only *level-one* node. The initial route between Node *F* and the controller is $F \rightarrow G \rightarrow D \rightarrow A \rightarrow$Controller. Assume that Node *F* runs the Authentic Neighbor and Shorter Path Discovery Phase and gets multiple responses from its neighbors. Among these responses, suppose that neighbor Node *B* has the shortest path, which is $B \rightarrow A \rightarrow$Controller. Thus, it is profitable for *F* to change the route to $F \rightarrow B \rightarrow A \rightarrow$Controller. However, the link between *F* and *B* is not secure. In order to secure this link, Node *F* sends a new key, K_{BF}, to Node *B* through the path $F \rightarrow G \rightarrow D \rightarrow A \rightarrow$Controller$\rightarrow$A$\rightarrow B$. First, Node *F* encrypts K_{BF} with its key shared with the controller. The controller decrypts K_{BF} and encrypts it with the key it shares with *B*. This method eliminates link-by-link encryptions and decryptions. Moreover, intermediate nodes do not learn K_{BF}. Having obtained K_{BF}, Node *B* responds with an authentic acknowledgment, $E_{K_{BF}}$ *(acknowledgment)*.

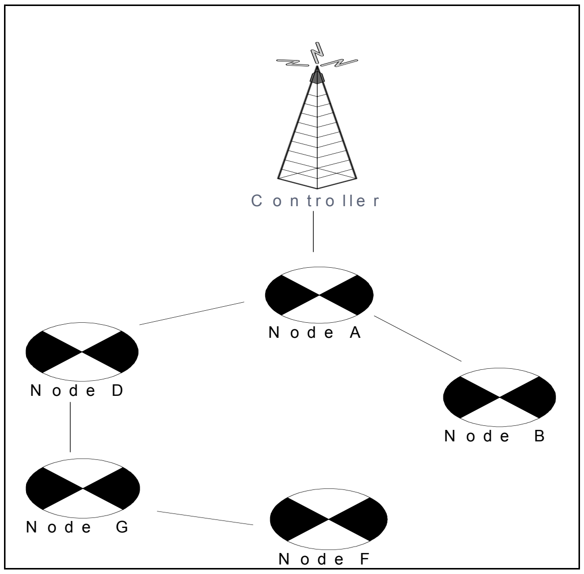
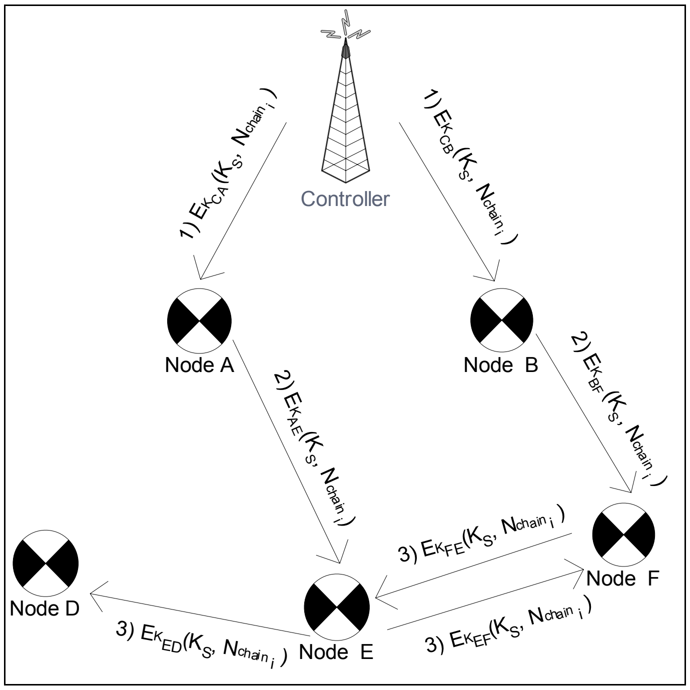


Fig. 6. Key Exchange Phase

Furthermore, it may be argued that if, for example, *D* is not happy about *B* trying to change the route, it can drop the key exchange packets. However, Node *F* will eventually realize this situation since it would not receive the authentic acknowledgment. In such a case, Node *F* can try other secure paths to perform this key exchange. Having only one secure path means very low cryptographic connectivity, which is very unlikely as stated in [1]. That is why there actually are several secure but longer paths between a node and controller although only one secure path is shown in Figure 6 for the sake of simplicity.

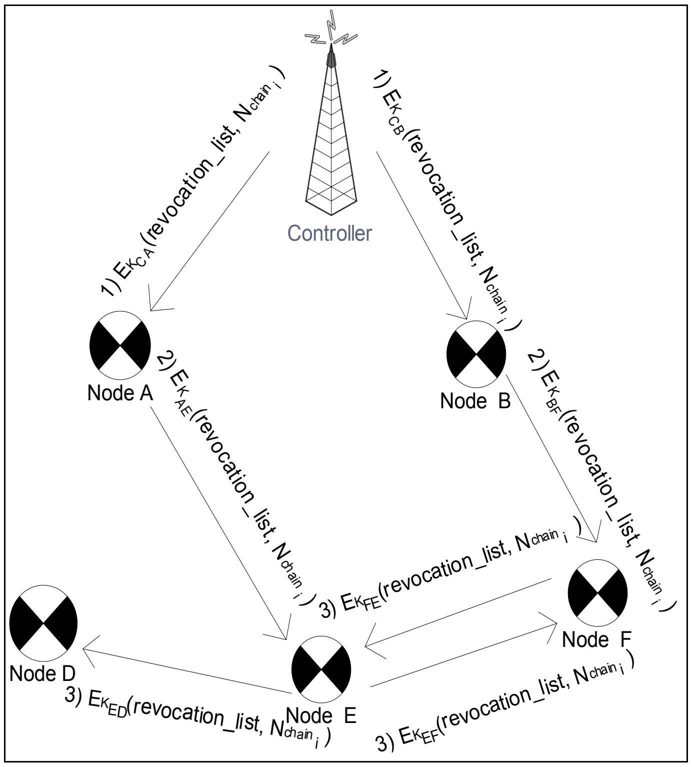
E. *Session Key Expiration Phase*

The controller may decide to invalidate a session key, before the normal lifetime indicated in the timestamp. Thus, the controller sends a session key expiration message to level-one nodes. The session key expiration message contains the invalidated session key and the next nonce of the nonce chain. Each node forwards the expiration message to its neighbors until all nodes in the network have the expiration message. The session key expiration messages are encrypted and decrypted with the key shared between the originator and the receiver of this message. Each node receiving the session key expiration message checks the validity of the next nonce of the nonce chain in order to authenticate that the message is originated from the controller. Figure 7 shows an example of the Session Key Expiration Phase. In this example, Nodes *A* and *B* are level-one nodes. Each node receiving this message decrypts it with the appropriate key, expires the session key, encrypts the message and forwards it over all of its secure paths. The protocol for distributing a new session key is similar to the Route Learning Phase except that the route field is not included in the message. Thus, the session key is simply distributed over the existing secure links.


Controller
1) $E_{K_{CA}}(K_S, N_{chain_i})$
1) $E_{K_{CB}}(K_S, N_{chain_i})$
Node A
Node B
2) $E_{K_{AE}}(K_S, N_{chain_i})$
2) $E_{K_{BF}}(K_S, N_{chain_i})$
3) $E_{K_{FE}}(K_S, N_{chain_i})$
Node F
Node D
3) $E_{K_{ED}}(K_S, N_{chain_i})$
Node E
3) $E_{K_{EF}}(K_S, N_{chain_i})$
Fig. 7. Example of Session Key Expiration Phase

F. Revocation Phase

If a sensor node is compromised, the key ring of that node must be revoked or deleted from the network. First, the controller sends the revocation list, i.e. the key identifiers of the revoked node, as well as the next nonce of the nonce chain to all of the level-one nodes, encrypted with the keys shared between the controller and the level-one nodes. Each sensor node receiving this message checks the validity of the nonce of the nonce chain. Second, all nodes forward this message to their neighbors until all nodes receive the message. Figure 8 shows an example of the Revocation Phase.


Controller
1) $E_{K_{CA}}(revocation\_list, N_{chain_i})$
1) $E_{K_{CB}}(revocation\_list, N_{chain_i})$
Node A
Node B
2) $E_{K_{AE}}(revocation\_list, N_{chain_i})$
2) $E_{K_{BF}}(revocation\_list, N_{chain_i})$
3) $E_{K_{FE}}(revocation\_list, N_{chain_i})$
Node F
Node D
3) $E_{K_{ED}}(revocation\_list, N_{chain_i})$
Node E
3) $E_{K_{EF}}(revocation\_list, N_{chain_i})$
Fig. 8. Example of *Revocation Phase*.

V. Attacks and Countermeasures

In this section, we briefly discuss possible attacks on the proposed protocol that a malicious node utilizes to harm the network and countermeasures that can be employed to thwart them. There exist two kinds of attackers: internal attackers and external attackers. An internal attacker tampers with a legitimate sensor node and learns all the keys saved in the device. An external attacker is an external node that gets to know only the session key. Attacker's aims are to eavesdrop on the data packets, to exhaust the battery of the nodes, and to inject false routing, session key expiration and revocation messages.

A. Route Poisoning Attacks

Assuming that an external attacker obtained the session key, the attacker could try sending false routes, as a reply to a shortest path discovery message in order to accumulate traffic over itself.

Countermeasure: This attack is not possible since the route in the reply packet is considered as the key exchange path, but since the route is false (non-existent route), the key exchange cannot take place. Eventually, a node will notice this false route during the key exchange protocol. This node could report this situation to the controller and thus invalidate the session key. Therefore, the nodes are obliged to reply with correct routes.

B. External DoS Attacks

The session key K_S is used for the Authentic Neighbor and Shortest Path Neighbor Discovery Phase. Each session key has a limited lifetime limited by the value indicated in the time stamp. When a legitimate node becomes compromised, the attacker may send a routing message (a Route Learning Phase message) with a very large session key expiration time in order to have sufficient time to build a denial of service (DoS) attack, where the objective is to exhaust the battery of its neighbors by repeatedly sending routing messages.

Countermeasure: In order to prevent this attack, each node must check the time stamp value. If it indicates a late expiration time, the node receiving this message must ignore the value and wait for a fixed amount of time for session key expiration. Therefore, the attacker has limited time to send multiple bogus shortest path discovery messages to exhaust the battery of sensor nodes.

C. Internal DoS Attacks

The use of the nonce chain limits the efficiency of certain types of attacks. For example, an unpublished nonce from the nonce chain is used by the controller to initiate a legitimate Route Learning, Session Key Expiration, or Revocation phase. Since the next nonce is known only by the controller, no other node can initiate a new phase. However, any internal attacker can modify the content of the message and send it through its secure links. For example, the internal attacker can turn a revocation message into a route-learning message to apply a DoS attack. The receiving nodes consider the message as legitimate.

Countermeasure: The nodes receiving the fabricated routing messages may also receive the legitimate revocation messages from other nodes; therefore, some nodes could receive two different types of messages with the same chain nonce. This situation can be reported to the controller as a security breach. The attacker can only deceive its downstream nodes if and only if it can prevent them from receiving the legitimate revocation message. This is only possible when the malicious node is the sole point of relay for its downstream nodes. However, this is unlikely. For instance, an example in [1] shows that if a DSN with 10000 nodes has a graph connectivity probability of 0.99999, on average, each node has 20 links.

D. Blackhole Attacks

The worst possible type of attack is called a blackhole attack, where an internal attacker who captured all the keys of a legitimate node is able to read and remove the messages of a subgroup of the network. Moreover, the attacker is able to change a legitimate route-learning message by modifying the route information. The aim of the internal attacker is to deceive the receiving nodes that its path to the controller is shorter; e.g. the attacker may claim that it is a level-one node. In this way, the attacker can accumulate traffic over itself, but it does not forward any data towards the controller. The evidence gathering method used in the previous situation is useless, since each routing message can naturally be different as the route and depth fields change at each hop, and there is nothing to compare for evidence. However, the attack is still local and the rest of the network is not compromised.

Countermeasure: In order to overcome the confidentiality problem, each node can send data encrypted with the key that it shares with the controller. Therefore, the attacker will not be able to read data packets but will be able to drop them. This problem can be solved by employing an authenticated acknowledgment mechanism after the route is established. In this mechanism, having received a data packet from Node, say, *X*, the controller creates an acknowledgment packet and encrypts it using K_{CX}, and sends it the Node *X*. Since, except Node *X*, only the controller can decrypt/encrypt using K_{CX}, if Node *X* receives the acknowledgment, this means not only the data packet has arrived, but also that Node *X* has a legitimate route to the controller and no node has dropped the packet en route. Authentic acknowledgments do not need to be requested all the time for blackhole detection; some random checks would help too. Failure in receiving one authentic acknowledgment would be considered as a weak indicator for a blackhole attack, since there might be other reasons behind the fact that the acknowledgment has not been received. However, it is quite clear that continuous failures strengthen the possibility of the blackhole attack. The correlation between the failure pattern and the existence of the blackhole attack is left as a future study.

VI. Simulations

We developed simulation software in Java to evaluate the performance of the proposed protocol. Important performance metrics considered are the overhead of the protocol in terms of routing and key exchange messages, and the average length of the paths connecting the sensor nodes and the controller. Our simulations aim at analyzing the changes in these metrics with respect to key ring size. In the simulations, we considered a network of 1000 nodes that are uniformly distributed and that show an average neighborhood connectivity of 40 nodes. The key pool has a size of 10000 keys.

Figure 9 depicts the change in the average path length from the sensor nodes to the controller before and after the authentic neighbor discovery phase respectively for key ring sizes. For small key ring sizes, the average path length is smaller when the authentic neighbor discovery phase is employed. This is quite natural considering that in this phase the nodes search for shorter paths to the controller. However, as the key ring size increases, the gains due to the authentic neighbor discovery phase diminishes, because the shortest path to the controller is also determined by the communication range of sensor nodes. Note that sensor nodes are limited in resources such as memory and transmission power. Thus, in order to increase the efficiency of data collection, the authentic neighbor discovery phase should be employed.

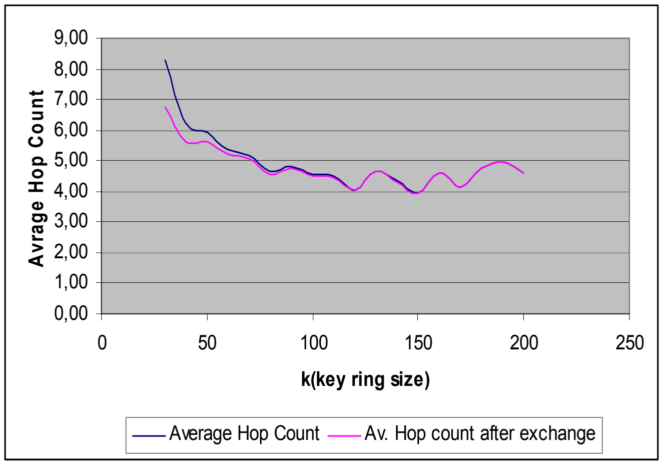


Fig 9. Average Hop Count with and without authentic neighbor discovery

Figure 10 depicts the number of route messages with different key ring sizes. In the simulations, we considered flooding as the method to exchange routing messages among the nodes. As the key ring size increases, more nodes can communicate securely and thus more nodes exchange routing messages to find a shorter path to the controller. Meanwhile, as the key ring size increases, the number of key exchange messages drops for the same reason outlined above.

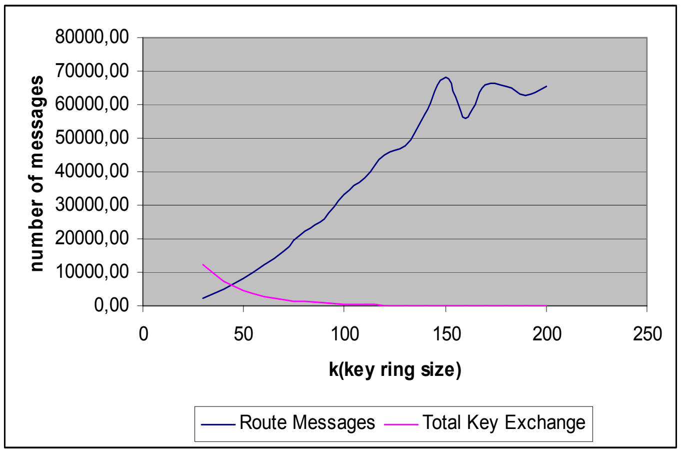


Fig 10. Key ring size versus number of messages

As an extreme case, where every node can communicate with all of its neighbors securely, the average path length reduces to 3.94 hops, but the total number of routing messages increases to approximately 70000. A similar average path length can be achieved by limiting the key ring size to approximately 100 thus reducing the average number of routing messages to approximately 40000.

VII. Conclusions and Future Work

In this paper, we propose a secure routing protocol for sensor networks. This protocol establishes secure routes in which links are secured using different keys. Thus, the compromise of a single key does not compromise the entire network.

The key idea behind the proposed scheme is random key pre-distribution which is also employed in [1]. Random key pre-distribution requires different nodes to be manufactured with a set of random keys selected from a pool of keys. When these nodes are deployed in the sensor field, the ones that are physically in the transmission range of each other will be able to communicate securely if they, by chance, share a common key. Otherwise, secure routes may still be formed but they may be longer. The proposed protocol allows the nearby nodes, which do not have common keys, to exchange keys, but only when this key exchange would yield a shorter and secure path.

The proposed protocol is flexible such that it allows a tradeoff between route length and the route setup cost in terms of processing power and storage. For instance, when there are stringent limitations on the memory space, the key ring size is small. Then, the secure routes are established over a longer physical path since the possibility of two nodes having a common key is low.

The simulations demonstrate that increasing the key ring size beyond a certain value does not have any effect on the average path length. Given the number of nodes in the network, an optimal key ring size can be found. The Authentic Neighbor and Shorter Path Discovery Phase will improve the average path length for small key ring sizes. As future work, some threshold schemes may be implemented in order to decide whether shorter path discovery is required.

Acknowledgement

This work was in part supported by Scientific and Technical Research Council of Turkey (TUBITAK) under the project number 104E007.

References

1. L. Eschenauer, V. D. Gligor, "A key-management scheme for distributed sensor networks". Proceedings of the 9th ACM conference on Computer and Communications Security, 2002, Washington, DC, USA
2. A. D. Wood, J. A. Stankovic, "Denial of Service in Sensor Networks", IEEE Computer, 35(10): 54-62, 2002.
3. J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing robust and ubiquitous security support for mobile ad-hoc networks," in ICNP, 2001, pp. 251–260.
4. J. P. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in mobile ad hoc networks," in Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001), 2001.
5. L. Zhou and Z. Hass, "Securing ad hoc networks", IEEE Network Magazine, volume 13, no 6,1999.
6. H. Luo, P. Zefros, J. Kong, S. Lu, and L. Zhang, "Self-securing ad hoc wireless networks," in Seventh IEEE Symposium on Computers and Communications (ISCC '02), 2002.

7. J. Binkley and W. Trost, “Authenticated ad hoc routing at the link layer for mobile systems,” Wireless Networks, vol. 7, no. 2, pp. 139–145, 2001.
8. J. Kong, H. Luo, K. Xu, D. L. Gu, M. Gerla, and S. Lu, “Adaptive security for multi-layer ad-hoc networks,” Special Issue of Wireless Communications and Mobile Computing, Wiley Interscience Press, 2002.
9. Y.-C. Hu, D. B. Johnson, and A. Perrig, “SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks,” in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), June 2002, pp. 3–13.
10. S. Basagni, K. Herrin, E. Rosti, and D. Bruschi, “Secure pebblenets,” in ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001), October 2001, pp. 156–163.
11. P. Papadimitratos and Z. Haas, “Secure routing for mobile ad hoc networks,” in SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), January 2002.
12. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, “SPINS: Security protocols for sensor networks,” in Proceedings of Mobile Networking and Computing 2001, 2001.
13. M. Tatebayashi, N. Matsuzaki and D.B.J. Newman, Key distribution protocol for digital mobile communication systems, in: Advances in Cryptology – Crypto’89, Lecture Notes in Computer Science, Vol. 435 (1989) pp. 324-334.
14. C. Boyd and A. Mathuria, Key establishment protocols for secure mobile communications: A selective survey, in: Australasian Conference on Information Security and Privacy (1998) pp. 344-355.
15. D. Liu and P. Ning. “Efficient distribution of key chain commitments for broadcast authentication in sensor networks”, In Proceedings of Network and Distributed System Security Symposium Conference, San Diego, 2003.
16. R. Di Pietro, L. V. Mancini, and S. Jajodia, “Providing secrecy in key management protocols for large wireless sensors networks”, Journal of Ad Hoc Networks, 1(4): 455-468, November 2003.
17. W. Du, J. Deng, Y.S. Han, P. K. Varshney, “A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks”, 10th ACM Conference on Computer and Communications Security, 2003, Washington, DC, USA
18. D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks", Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, October 27-31 2003, pp. 52-61.
19. S. Zhu, S. Setia and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), Washington, DC, USA, October 27-31 2003.