Post-deployment Based Key Distribution Mechanism for Wireless Sensor Networks

Qiuhua Cao and John A. Stankovic
Department of Computer Science
University of Virginia
Charlottesville, VA, 22903
{qhua,stankovic}@cs.virginia.edu

***Abstract*— To secure the communications in wireless sensor networks, sensor nodes have to obtain secret keys. Many key management schemes, such as KDC and asymmetric cryptography, are not suitable for wireless sensor networks due to the strict resource constraints. Most key pre-distribution algorithms require that each node stores a set of keys before the deployment and discovers the keys shared with the neighboring nodes after the deployment. In this paper we present an efficient key distribution algorithm based on the topology and the secure communication requirements of the system. Our algorithm assigns keys to each node after the deployment. Thus, each node needs to store sufficient number of keys only. By providing the mechanism to secure the key distribution process, our algorithm enables the system to recover from severe attacks by redistributing the keys. Based on the simulation results and the analytic study we demonstrate that our algorithm is efficient in memory, bandwidth, and energy. At the same time, our algorithm is able to provide 100% connectivity and stronger resilience to node capture.**

I. Introduction

Wireless sensor networks have been widely studied and applied to many applications in military as well as civilian operations. Many applications are dependent on secure operations, such as key management, to secure the communication between sensor nodes in a network. Otherwise, secret information can easily be accessed by adversaries. For example, an intruder can violate security of the system by interfering with system availability, data integrity, or data confidentiality. In order to defend against adversaries, node-to-node communication in wireless sensor networks should be encrypted and authenticated. However, encryption and authentication require that the nodes in the system share secret keys, which brings up one main question "how to set up secret keys between communicating nodes?".

There are three types of key management mechanisms in the literature providing the answers to the question above. The three types are key distribution center (KDC) mechanisms, asymmetric cryptography mechanisms, and key pre-distribution mechanisms. However, KDC [16] and asymmetric cryptography [5] [18] are computation, bandwidth and memory inefficient for the highly limited devices found in wireless sensor networks.

Commonly, a key pre-distribution strategy such as the algorithm proposed in [10] is composed of three steps: (1) generating a pool of P random keys, (2) each node randomly selecting a set of k distinct keys from the pool and storing the keys in memory before deployment, and (3) discovering one shared node-to-node key after deployment. The result is that the pre-distribution key management schemes are not very efficient. For example, the pre-deployed key scheme in [10] requires k^2 decryptions on the receiver side, k encryptions on the sender side, and at least k messages to be sent and received for the key discovery procedure to find which keys are actually shared even for the nodes in their communication ranges. It is also memory inefficient to store k distinct keys.

The first observation we have is that the inefficiency of pre-distribution key algorithms is because no a priori knowledge of deployment configuration of the system is available at the time of assigning the keys. A node is deployed randomly and it can not know beforehand which nodes will be within its communication range after the deployment. As a result, a node has to preload a subset of sufficient keys before deployment to satisfy the connectivity requirement of the system after deployment. Even if a node is deployed by hand, it is also costly to pre-determine the location of each node.

The second observation we have is that many key pre-distribution and management algorithms do not take the secure communication requirements of an application to which the keying algorithms apply into account. For example, some applications such as tracking require the nodes in a neighborhood or a group to cooperate with each other, while some applications such as environ-

mental monitoring only need the nodes in the network to take a sample and send the data back to the base station periodically. The first case requires a node to have pairwise keys to share with nodes in a neighborhood or a group to build secure communication among them. The latter case only requires the secure communication between a node with its communication partner (i.e., its parent).

The third observation we have is that many key pre-distribution and management algorithms ignore what if the keys of the system are system-wide exposed after severe attacks.

Driven by the three observations above, we propose an efficient self-healing post-deployment based key distribution mechanism for wireless sensor networks. The basic idea of our key distribution mechanism is to distribute the keys based on post-deployment knowledge and on the secure communication requirements of the system. In this way, we eliminate the need for each individual node to store a large subset of preloaded keys and we also reduce the communication cost to discover keys. At the same time, our algorithm gives the system the flexibility to redistribute keys when necessary, such as after severe attacks, to recover the system.

The contributions of our algorithm are:

- providing an efficient key distribution mechanism for wireless sensor networks based on post-deployment knowledge of the system,
- providing an efficient mechanism to heal the system from attacks by rekeying the system, and
- providing flexible keying mechanisms based on security communication requirements of the system.

We organize the rest of the paper as follows. We discuss the related work in Section II. We give the overview of our algorithm in Section III. In Section IV, we present the trust model and attack models addressed in this paper. We discuss the details of our algorithm in Section V. The analyses of our algorithm with respect to the attack models addressed in this paper are presented in Section VI. In Section VII, through simulations we demonstrate that our algorithm is efficient. We also compare our algorithm with other works by analytic analyses. We conclude our paper in Section VIII.

II. Related Work

Recently, several other research groups [11] [15] [9] [3] further investigated how to make public key certificates, such as algorithms based on Elliptic Curve Cryptography (ECC), usable in wireless sensor networks. Despite the communications and computational reductions provided by the ECC technology, interactive key management protocols, even using faster and smaller public key algorithms, incur significant bandwidth and large latencies when used in wireless sensor networks.

Random pairwise key pre-distribution algorithms assign a set of keys to a node based on probability or random graph theories [10] [4] [17] or symmetric matrices operations [1] [7] [6] or polynomial computations [2] [13] before deployment. The random pairwise pre-distribution algorithms reduce the memory requirements to store the preloaded keys. However, the random subset assignment algorithms need to use correct parameters to guarantee that any two nodes can establish a pairwise key, provided that the two nodes can communicate with each other. At the same time, subsets of keys have to be sufficiently disjoint from one another. Otherwise sufficient enough node captures can result in the exposure of all or a large fraction of the keys in the system. These two requirements demand a large subset of keys to be assigned to each node. And the size of a subset of keys a node has to store dramatically increases when the network size increases.

Several papers proposed ideas to reduce the memory requirements of random pairwise pre-distribution algorithms by exploiting the location/deployment information of the nodes in static wireless sensor networks. [14] integrated the location information with both a random pairwise key pre-distribution scheme and a polynomial based key predistribution scheme. [8] developed a key pre-distribution scheme based on the model of the node deployment knowledge. The spatial relation between nodes derived prior to deployment are used to assign the keys to a node. The authors of the paper modeled the node deployment knowledge using a non-uniform probability density function, normal distribution. Similarly, [12] proposed a grid-group deployment scheme. In [14] [8] [12], the location/deployment knowledge of a node is an estimation of which area a node is to be deployed. The results indicate that the location or deployment knowledge is useful to avoid unnecessary key assignments in pre-distribution algorithms. However, the performance of the algorithms is highly influenced by the difference between the prior location/deployment knowledge of a node and the real location of the node. And to guarantee the connectivity between the nodes in the adjacent zones/groups/grids is difficult.

Different from the algorithms above, our key distribution algorithm assigns keys to nodes after the deployment. In this way, the keys are assigned to nodes which are really adjacent. Our algorithm further reduces the memory requirement of a node to store the keys. At the same time, our algorithm establishes sufficient keys for any nodes which require communication.

III. Overview of Our Algorithm

An important underlying idea of our key distribution algorithm which contributes to its efficiency is to distribute the keys to the nodes in the system after the deployment. The algorithm operates in 4 phases. First, the base station issues a topology discovery message. Second, upon receiving the topology discovery request, each node reports its topology (neighbor) information to the base station. Third, the base station constructs the system topology based on the reports from the network. Fourth, the keys are generated by the base station knowing the system topology and system communication requirements. For example, the communication requirements might be that all nodes must communication with neighbors, with the base station and perhaps there is also group communication. In this case all pair-wise keys, keys between nodes and the base station and group-wise keys can all be generated by the base station and disseminated properly. Our algorithm also enables the base station to redistribute new keys to the nodes in the system, whenever it is necessary, to enable the system to recover from attacks. The redistribution of the keys can be done by reassigning and distributing keys to the nodes without rebuilding the topology, or it can be done from rebuilding the topology of the system. Figure 1 shows the steps of our algorithm. Topology discovery, topology reporting and topology construction can be piggybacked with typical system initialization schemes found in most wireless sensor networks so the cost of our algorithm in these three phases is minimum.

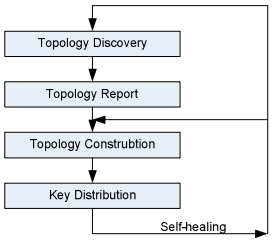


Fig. 1. Overview of Our Algorithm

As discussed in section I, our algorithm provides an answer to the question "how to set up secret keys between communicating nodes?". Our algorithm also provides the answer to the question"how to heal the system from security attacks where keys are divulged" by rekeying the system. In the remaining part of this section, we discuss how our algorithm makes the key set-up process secure (with respect to the attack models addressed in this paper) in a general manner. The detailed algorithm is discussed in Section V. In order to make sure that the key set-up process is secure, each step of our algorithm is protected. Our algorithm makes sure that each node can correctly authenticate the topology discovery request from the base station. In the topology reporting phase, each node uses an unique key to encrypt the information before sending it back to the base station and only the base station can correctly decrypt the information. The base station verifies the correctness of the reported topology information and authenticates the source of the reported information during the topology construction step. During key distribution, the key information is encrypted and only the receiver of the key information can decrypt it.

IV. Trust Model and Attack Models

Before the discussion of the details of our algorithm, we discuss the trust model and attack models addressed in this paper.

A. Trust Model

We assume that a PC is the trusted center as the base station. Before deployment, the base station generates a sequence of authentication codes $c(1)$, $c(2)$, ..., $c(k)$, ..., $c(n)$, where $c(k+1)$=$F(k)$. F function is a one-way function which is computationally infeasible to compute $c(k-1)$ in a limited time by knowing $c(k)$ and F. Also before deployment, we assume that each node stores (1) an unique shared secret encryption key K_{encr} with the base station, (2) an unique shared secret authentication key K_{mac} with the base station, (3) the function F, and (4) the value $c(n)$ in its memory. We also assume that an adversary has the same communication capability as the wireless sensor devices in the system. At the same time, we assume that only one adversary is in a neighborhood.

B. Attack Models

Eavesdropping: An adversary could easily gain access to private unencrypted information by monitoring the wireless transmissions between nodes. Therefore, when a node reports the topology information back to the base station, we use end-to-end encryption to defend against the eavesdropping at this phase. When the base station sends the key information to the nodes in the network, we also use end-to-end encryption.

Spoofing, Altering, Replaying: An adversary can spoof or alter or replay an overheard message being transmitted between the nodes in the network if the message is not encrypted properly. However, our algorithm utilizes the implicit authentication properties of a message to confine the effect of spoofing or altering or replaying. Thus, our

algorithm requires minimum number of predeployed encryption keys. To defend against replaying, we maintain the freshness of an authentication code.

DOS: Denial-of-Service(DOS) attacks aim to destroy network availability. Attackers can send a series of meaningless communications causing the targeted nodes to exhaust their batteries while processing and forwarding the messages. Proper authentication can prevent injected messages from being accepted by the network. For example, using signatures based on asymmetric cryptography can provide message authentication. However, this technique is highly computationally intensive. In our approach only the base station is allowed to send any flooding messages thus mitigating the DOS attack. In addition, symmetric authentication is used to prevent any injected flooding messages from being propagated through the network.

V. Our Efficient Key Distribution Algorithm

In addition to the trust assumptions in Section IV-A, we make one non-security related assumption, reliable communication. Later in the analyses section VI, we discuss the impact of this on our algorithm. Furthermore, we show that the assumption can be relaxed.

Our algorithm is organized in 4 phases: topology discovery, topology report, topology construction and key distribution.

1) Topology Discovery: At the beginning of topology discovery, the base station sends out a topology discovery message (TDM) as tuples <sender, authcode> to all the nodes in the system. The "sender" is the identity of the node to send/relay the message. The "authcode" is the authentication code generated by the base station to be used for the receiver to authenticate if the message is sent by the base station. The authentication code is chosen from the sequence $c(1)$, $c(2)$, ..., $c(k)$, ..., $c(n-1)$. The first topology discovery message sent from the base station selects $c(n-1)$ as the "authcode", the second topology discovery message uses $c(n-2)$, the *k*th discovery message chooses $c(n-k)$, and so on. The different rounds of the topology discovery messages are used when the system needs to redistribute the keys for self-healing purposes. Each node in the network receives a topology discovery message to (1) authenticate if the topology discovery message comes from the base station, if not, the message gets dropped; (2) rebroadcast a topology discovery message with the same authentication code $c(j)$ one time; and (3) build up its neighbor table by overhearing the messages in its communication range.

Authenticate the Base Station: As described in the assumptions, we only allow the base station to initiate the topology discovery messages as flooding messages. An adversary is able to pretend to be the base station because the topology discovery messages are not encrypted at all (no key for encryption available yet). If no proper authentication mechanism is built in the system, a node is not capable of deciding whether it should respond to a received message or not. We use the one-way function F and the value $c(n)$ (note that F and $c(n)$ are stored on each node before deployment) to authenticate if a topology discovery message is sent by the base station. The authentication process is as follows: when a node in the network receives a topology discovery message with "authcode" $c(j)$, it computes the value $c(j')$=$F(c(fresh))$ while the $c(fresh)$ is the most up-to-date authentication code the node received from the base station. Initially, the $c(fresh)$ is set to be $c(n)$. And if the computed $c(j')$ equals the $c(j)$ received, the message is generated by the base station and is authenticated. Otherwise, the message is dropped. The authentication algorithm above guarantees that only a new topology discovery message initiated by the base station with a newer authentication code (newer than $c(fresh)$) is accepted by the authentication algorithm. Any old messages (messages with older authentication codes, with respect to $c(fresh)$) can not get authenticated.

Broadcast a Topology Discovery Message: Once a topology discovery message is authenticated, a node which receives/overhears the message (1) sets its most up-to-date authentication code value $c(fresh)$ to be $c(j)$, (2) broadcasts a topology discovery message (only the sender of the authenticated message changed to be the identity of the node broadcasting this message) given that it is its first time to receive the message with "authcode" $c(j)$. In this way, each node only broadcasts a topology discovery message with the same "authcode" once.

Build Up the Neighbor Table: When a node receives/overhears a topology discovery message, if the received message is authenticated as above, the node records the sender of the authenticated message in its neighbor table if the identity of the sender is not yet in the table.

2) Topology Report: After the topology discovery phase, each node has the most up-to-date authentication code $c(fresh)$ from the base station and a table of neighbor identities. During the topology report phase, each node reports its neighbor information back to the base station. The topology report message (TRM) is a 4-tuple <source, dest, authcode, SECRET >. The "authcode" is filled with the most up-to-date authentication code $c(fresh)$ a node received from the base station in the topology discovery phase. the "SECRET", as given below in Equation (1), is the encrypted neighbor table

```
% Filter False Messages
For each TRM of all received TRMs
    if(SECRET decryptable and authenticated)
        add source → NodeList; add nt → nt_source;
    elseif(SECRET not authenticated || not decryptable)
        add source → FalseNodeList; drop the message;
% Sanitize Neighbor Tables
For each nt_j(j = 0, ..., k)
    for each N_i ∈ nt_j
        if N_i ∈ FalseNodeList {delete N_i from nt_j;}
```

Fig. 2. Neighbor Table Sanitizing Algorithm

(nt), source, and authentication code using the key K_{encr} shared between a node and the base station, and the message authentication code (MAC) generated using the authentication key K_{mac}.

$$tpinfo = \{source, authcode, nt\}$$
$$SECRET = \{tpinfo\}_{K_{encr}}, MAC(K_{mac}, tpinfo_{K_{encr}})$$

When a node receives a topology report message as the chosen receiver, the chosen receiver is decided by the routing algorithm of the system, it relays the message when (1) the "authcode" in the message is legitimate (equals to the most up-to-date authentication code), and (2) the "dest" of the message is the base station. Otherwise, the message is dropped.

3) Topology Construction: From the discussions above, we know that at the end of the topology report phase, the base station received a list of neighbor tables encrypted as in Equation (1) from the nodes in the network. In the section, we discuss the topology construction algorithm.

Before the base station constructs the topology of the network, it is vital that the neighbor tables to be used to construct the topology are legitimate. We provide a neighbor table sanitizing algorithm for the base station to filter out malicious topology report messages, get rid of adversaries in neighbor tables. In figure 2, we present the pseudo algorithm for processing the neighbor tables collected from the network. The nt_{N_i} in figure 2 represents the neighbor table of node N_i.

First, the neighbor table sanitizing algorithm builds the legitimate neighbor tables nt_{N_i}s and the source of the legitimate messages $NodeList$. And it filters out the false messages (which are not decryptable or can not pass the authentication because the adversary does not have the correct shared keys) sent by the adversary. The sources of the undecryptable or not authenticated messages are added to the list $FalseNodeList$. Second, the algorithm sanitizes the neighbor tables nt_{N_i}s against the false node list.

After the sanitizing of the neighbor tables, the base station uses the legitimate neighbor tables nt_{N_i}s and the $NodeList$ to construct the topology of the system, which is to construct a graph G(V,E). All nodes in $NodeList$ are vertices (V). If a node is in a vertex's neighbor table, there is an edge (E) between the node and the vertex.

4) Key Distribution: Once the topology of the network is constructed, the base station: (1) generates a sufficiently large key pool using any of the key generation algorithms in the literature; (2) assigns proper keys for the nodes which need to communicate with each other based on the secure communication requirements of the system; (3) disseminates the assigned keys to the corresponding nodes in the network, the keys are properly encrypted so that only the receiver of the keys can obtain the keys. In the remaining part of this section, we discuss the details of the key assignment algorithm and the key dissemination algorithm.

The key assignment algorithm at step (2) gives the base station the flexibility to assign different keys (i.e. pairwise key, single key, group key, parent-child key) for different purposes based on the secure communication requirements and the topology of the system. In this paper, we discuss two kinds of keying mechanisms, namely spanning tree based and neighborhood based. The spanning tree based keying mechanism provides the keys for some applications (i.e. environmental monitoring, agriculture, medicare), where nodes in the network do not require cooperation among neighbors and use static routes (i.e. spanning tree) to transmit messages. Each node in the network needs to directly share its own information with its parent, and a parent needs to share its information with its children. The neighborhood based keying mechanism provides the keys for the applications (i.e. military tracking), where nodes in a neighborhood are required to share information and cooperate with each other to accomplish a task.

Spanning Tree Based Keys ($STB-K$): Each node has a maximum of two keys (leaf nodes only have one key) to be distributed, one key is used to communicate with its parent, one key is used to communicate with its children. The STB-K algorithm starts from the base station as shown in Figure 3. First, the base station (BS) selects an unique key to be shared between itself and its 1-layer children (node N_1, node N_2, node N_3). Second, the base station selects an unique key to be shared between each of nodes in the i-layer and its children in the $i+1$-layer. For example, node N_2 in the 1-layer shares an unique key with node N_4, node N_5, and node N_6 in the 2-layer. And third, the base station repeats the second step until the bottom of the tree. In Figure 3, the same color edges/links share the same unique key.

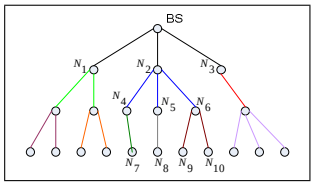


Fig. 3. An Example of the STB-K Algorithm

$i = 0$; $While(i < sizeof(V))$ {
$v = V_i$; $vs_v = \phi$; $vk_v = \phi$; $j = 0$; $i = i + 1$
for each $x \in V$; *if* $e(v, x) \in E$ {
$x \rightarrow vs_v$; $x \rightarrow vk_v$; *delete* $e(v, x)$ *from* E };
$While(j < sizeof(vs_v - 1))$ {
$v_a = vs_v[j]$; $v_b = vs_v[j + 1]$; $j = j + 1$
if $e(v_a, v_b) \in E$ { *delete* $e(v_a, v_b)$ *from* E;
if $v_a \notin vk_v$ { $v_a \rightarrow vk_v$; }
if $v_b \notin vk_v$ { $v_b \rightarrow vk_v$; } } }
assign an unique key for all the vertices in vk_v; }

Fig. 4. Pseudocode of the NBB-K Algorithm

Neighborhood Based Keys ($NBB - K$): Each node has K keys, K is defined by the difference between W (number of neighbors) and Q (number of neighbors that share the same neighbor with this node), plus one. The $NBB - K$ algorithm assigns keys based on the system topology $G(V, E)$. The pseudocode of the NBB-K is as shown in Figure 4.

For the key dissemination at step (3), the base station can simply unicast the keys to individual nodes. However, a more energy and bandwidth efficient way to disseminate the keys is to piggyback the keys of several nodes together, when the base station has the static routing information of the system (i.e. spanning tree). The keys are properly encrypted. Here, we use the spanning tree as the exemplary static routing algorithm to illustrate how to piggyback the encrypted keys of several nodes together. The two dissemination mechanisms are:

Spanning Tree Based Dissemination ($STB - D$): The encrypted keys of the nodes on the same sub-tree are piggybacked together. In Figure 5, We use the subtree $BS \rightarrow N_2 \rightarrow N_4 \rightarrow N_7$ as shown in Figure 3 as the example to demonstrate how the STB-D algorithm properly encrypts and disseminates the keys of the nodes on one sub-tree. The key assignment algorithm in the example is the STB-K. $K_{(BS,N_i)}$ in Figure 5 represents the K_{encr} shared between the base station and node N_i. The key dissemination message is composed of two parts, separated by a special byte. The first part is the encrypted node identities of the receivers of the keys, the second part is the encrypted keys. As an example, in Figure 5 the base station sends message KDM_{N_2} to node N_2. The encrypted node identities are generated as follows: the identity N_7 is first encrypted using key $K_{(BS,N_7)}$; then it combines with identity N_4; finally the identity N_4 and the encrypted N_7 are encrypted using key $K_{(BS,N_4)}$. P_k and C_k represent the key for a node to communicate with its parent and the key to communicate with its children, respectively. The keys for node N_i are encrypted using key $K_{(BS,N_i)}$. For the key assignment algorithm NBB-K, we replace the encrypted P_k and C_k with the properly encrypted neighborhood based keys. Upon receiving key distribution information, a node processes and relays the message to its next hop using the algorithm shown in Figure 6.

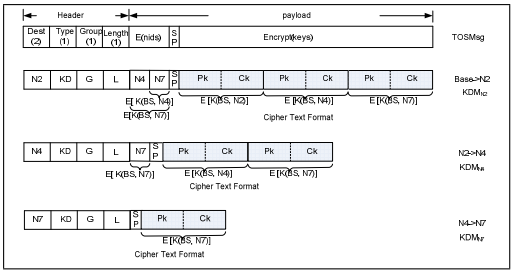


Fig. 5. The Key Distribution Message Format for $\{STB - K, STB - D\}$

Unicast Based Dissemination (UCB-D): The encrypted keys for each node are sent out individually by the base station.

VI. Analysis

We addressed three types of attack models in section IV. In section V, we presented the details of our key distribution algorithm. In this section, we first generate a complete list of possible attacks at each phase of our algorithm, then we demonstrate that our algorithm correctly defends against the attacks. Later, we analyze the impact of the assumption on the reliable communication on our algorithm in the connectivity analyses, and show how the assumption can be relaxed.

A. Correctness demonstration of our algorithm against the attack models

As stated in section V, our algorithm is composed of 4 phases. And the adversary is capable of eavesdropping and spoofing/alerting/replaying the messages during the 4 phases if the messages are not encrypted properly. A topology discovery message (TDM) in the topology discovery phase is a 2-tuple <sender, authcode>. A topology report message (TRM) in the topology report phase is a 4-tuple <source, dest, authcode, SECRET>. The topology construction phase is the computation done

```
% Upon Receiving a KDMsg
if(I am the dest) % receives a KDMsg
    payload = KDMsg.payload; Index(SP) = payload[SP];
    if(I am not the last hop) % relays a KDMsg
        decrypt KDMsg.payload[SP + 1] using K(BS, ME);
        save the decrypted keys;
        send a KDM to the next hop;
    else(I am the last hop of the KDMsg)
        decrypt KDMsg.payload[SP + 1] using K(BS, ME);
        save the decrypted keys;
else(I am not the dest) % routes a KDMsg
    route the KDMsg to next hop;
```

Fig. 6. Pseudocode of the Algorithm to Receive, Relay, and Route a KDMsg

on the base station. A key distribution message (KDM) in the key distribution phase is encrypted properly as shown in Figure 5, all the fields of the payload are end-to-end encrypted using the shared secret key K_{encr} between the base station and the receiver of the message. The result is the only two phases an adversary can attack are the topology discovery and the topology report phases.

Attacks in the Topology Discovery Phase: There are four cases an adversary can attack the topology discovery process after overhearing the legitimate messages. Case (1), the adversary eavesdrops on a legitimate authcode "c(j)", then forges a TDM <adversary, c(j)>. Case (2), the adversary forges a TDM <sender, spoofed authcode>. Case (3), the adversary forges a TDM <adversary, spoofed authcode>. Case (4), the adversary replays an older authcode "c(k)" in a TDM <sender, c(k)>, where c(k) is the legitimate authentication code used in the previous round of the topology discovery phase. In case (2), case (3) and case (4), the forged TDM or replayed TDM gets dropped in one hop because the spoofed or the older authentication code is not able to pass the authentication algorithm as discussed in section V-.1. In case (1), the identity of the adversary is possibly recorded in the neighbor table of a legitimate node when it builds its neighbor table. However, in the topology construction phase in section V-.3, our neighbor table sanitizing algorithm as shown in Figure 4 deletes the adversary from the legitimate neighbor tables. So we conclude that our algorithm defends against the attacks in the topology discovery phase correctly.

Attacks in Topology Report Phase: After eavesdropping on the TRMs in the topology report phase, an adversary is able to manipulate the 4 fields of a TRM. However, the manipulation on the "dest" and/or the "authcode" does not affect the correctness of the algorithm. As discussed in section V-.2, one condition for a receiver of a TRM to relay the message is that the "dest" has to be the base station, because all the TRMs are reported to the base station. A legitimate "authcode" is another condition for a receiver to relay a TRM. The possible attacks are as follows: Case (1), by faking the "source", an adversary issues a TRM <adversary, dest, authcode, SECRET>; Case (2), an adversary forges a TRM <sender, dest, authcode, forged SECRET> or a TRM <adversary, dest, authcode, forged SECRET>. In case (1), the forged TRM does not harm the correctness of our algorithm because the SECRET are legitimate and the topology construction algorithm builds the topology of the system based only on the legitimate SECRET. It is just that the base station receives two identical SECRETs. In case (2), once the illegitimate TRM arrives at the base station, the message gets dropped because the SECRET is not decryptable or authenticated. To conclude, our algorithm defends against the attacks in the topology report phase.

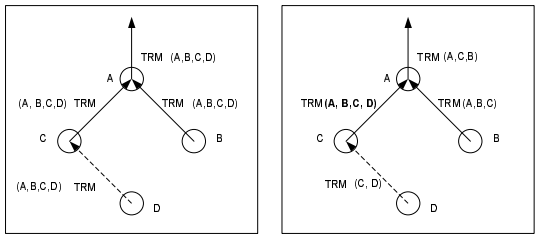
Based on the discussion above, we conclude that our algorithm correctly defends against the attacks as we addressed in the section IV.

B. Connectivity Analysis

Connectivity is defined as the probability that any two neighboring nodes share one key. To provide complete connectivity the topology constructed by the base station has to be complete (include all the nodes in the system) and a node has to receive one of the KDMs destined to it. As discussed in section V, our algorithm guarantees the completeness of the constructed topology and each node receives the keys destined to it under the assumption of the reliable communication. In the following, we discuss the impact of the message loss on the completeness of the constructed topology and the message loss of KDMs on the probability of a node to obtain the keys. The message loss in the topology discovery and report phases impacts the completeness of the constructed topology. We discuss the message loss in each phase.

The Impact of Message Loss in the Topology Discovery Phase: During the topology discovery phase, the base station floods TDMs to the network. Each node broadcasts TDMs and updates its neighbor table when it receives a legitimate TDM. Hence, a node is isolated from the rest of the network at this phase only under one extreme condition when it is unable to communicate with any of the neighbor nodes. We argue that the isolation of a node is unlikely when all the nodes can communicate with their neighbors. But it is possible that a node is in some of the neighbor tables of it's neighbors but not in all of them.

The Impact of Message Loss in the Topology Report Phase: A node sends its topology (neighbor) information back to the base station during the topology report phase. The number of the TRMs a node sends is defined by the beacon rate and the period of the topology report phase. As long as one of the TRMs sent by a node arrives at the base station, the TRM contributes to the topology construction on the base station. The only case where the message loss may impact the completeness of the constructed topology is when none of the TRMs sent by a node reach the base station. Even if this is the case, it may still be possible to construct the correct topology. As shown in Figure 7(a), the message loss of all the TRMs sent by node D does not affect the completeness of the constructed topology because D is reported in the TRMs of nodes A, B, and C. Alternatively, as shown in Figure 7(b) the message loss of all the TRMs sent by a node can impact the completeness of the constructed topology when the neighbor tables of the neighbor nodes are incomplete. For example, in the constructed topology, node D is not the neighbor of node B.



(a) Case 1. (b) Case 2.

Fig. 7. Impact of Message Loss at the Topology Report Phase.

A node does not obtain the keys assigned to it only when all the KDMs destined to it are lost. To conclude, the message loss only impacts our algorithm in special cases, for example, when all the KDMs sent for a node are lost. By increasing the number of the transmissions of the TDMs, TRMs, and KDMs, the impact of the message loss on our algorithm becomes insignificant.

VII. Performance Evaluation

We evaluate our post-deployment based key distribution algorithm by simulation over different system deployments. Guided by the performance data obtained through simulations, we analytically compare our algorithm with two representative pre-distribution algorithms – *Eschenauer-Gligor Scheme* [10] and *Du-Deng Scheme* [8].

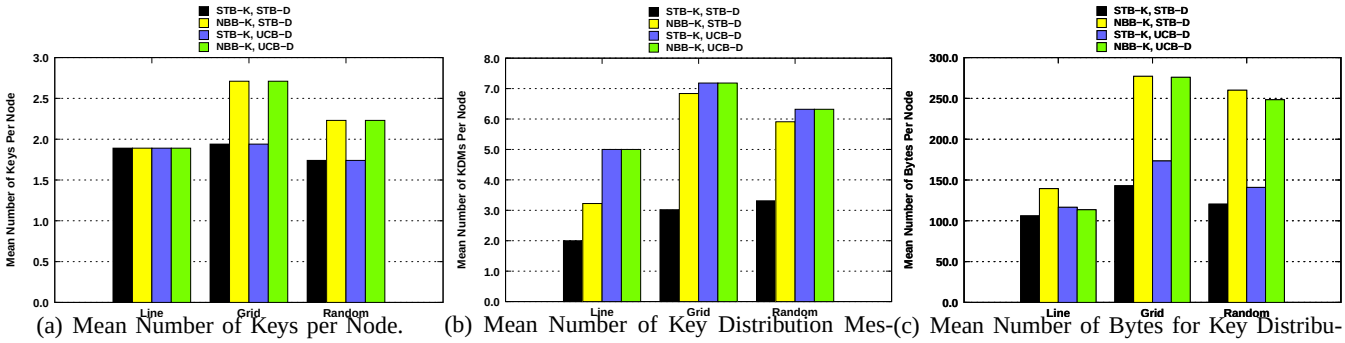
A. Simulation Results

In our simulation studies, we demonstrate the performance of our algorithm using the following metrics: memory, bandwidth and energy. The memory requirement for the keys per node is calculated from the number of keys multiplied by the number of bits per key. Bandwidth is measured by the number of bytes of the key distribution messages a node receives, relays, and routes. The energy consumption is represented by the number of messages a node processes. We study three system deployments – Line, Grid, and Random. In the Line deployment, we put all the nodes in a line, hop by hop. In the Grid deployment, we partition the deployment field into grids, the width of a grid equals the communication range. Each node is deployed at a grid point. In the Random deployment, we uniformly distribute the nodes in the deployment field. The purpose of the Line deployment is to study the impact of the STB-D and the UCB-D dissemination algorithms on bandwidth and energy under the condition that the STB-K and the NBB-K assigns the same number of keys to the nodes. The purpose of the Grid deployment is to compare against the Random deployment to understand the performance of our algorithm when systems have the same number of nodes, different deployment configurations, and yet the same key assignment and distribution algorithms. For both the Grid and the Random cases, the deployment field is 100m by 100m. The radio range is 25m. The number of nodes are 50. The length of a key is 56 bits. All the results are the mean over 10 runs. All the simulation results are within 5% of the mean and the confidence level is 95%.

Figure 8(a) demonstrates that the system topology and the secure communication requirements of the system define the number of keys per node. Each node is assigned sufficient number of keys. For the same system topology, the NBB-K results in more keys per node than the STB-K when nodes have neighbors other than their parent and children. The maximum number of keys per node assigned by NBB-K are defined by the W and Q as discussed in section V.

Figure 8(b) presents the mean number of KDMs a node receives, relays, and routes. The definitions of the three operations are as shown in Figure 6. In the rest of the section we rename $\{$STB-K, STB-D$\}$ with STB1, $\{$STB-K, UCB-D$\}$ with STB2, $\{$NBB-K, STB-D$\}$ with NBB1, and $\{$NBB-K, UCB-D$\}$ with NBB2 for clarity. Furthermore, we use a notation Topology:Algorithm to simplify the discussion. For example, L:STB1 stands for Line topology for STB1 algorithm.

The number of key distribution messages depends on the number of keys per node, the topology of the system, and the key distribution algorithm. This is illustrated by the results in the Figure 8(b). For instance, STB1 shows the best performance in the number of KDMs because



(a) Mean Number of Keys per Node. (b) Mean Number of Key Distribution Messages per Node Receives, Relays. (c) Mean Number of Bytes for Key Distribution Messages per Node Receives, Relays, and Routes.

Fig. 8. Mean number of Keys, Key Distribution Messages, and Bytes per Node

fewer keys are assigned per node and the dissemination algorithm packs key information for more nodes per message. The topology affects the number of KDMs. For example, L:STB1 requires fewer number of KDMs per node than G:STB1 because non leaf nodes route more KDMs in the Grid deployment than in the Line deployment. STB1 requires fewer KDMs per node than NBB1 for the same topology becaue NBB1 piggybacks key information for fewer nodes. The figure shows that L:STB1 performs 50% better than L:NBB1, while both experiments have the same topology and the key dissemination algorithm. The difference in performance is due to the key assignment algorithm. L:NNB1 has more specific information in KDMs including the node identities that are paired with the keys resulting in longer chunks of key information per node. Therefore, fewer chunks can be piggybacked to each message due to the message length constraint and a larger number of messages needs to be sent. Changing the topology for NNB1 from Line to Grid increases the number of KDMs, because in the G:NNB1 case the key information per node is larger due to more node neighbors. Therefore fewer chunks can be piggybacked per message and more messages need to be sent to distribute the key information. Since the topology change has less effect on STB1, G:STB1 performs 130% better than G:NBB1.

Figure 8(c) shows the mean number of bytes for KDMs per node. The number of keys is the same for all algorithms the case of Line topology, resulting in a comparable performance. However, for the Grid and Random topologies the NNB1 performs much worse than STB1. This is because the key information length per node is larger due to more neighbors and the overhead is higher for NNB1 in comparison to STB1.

Based on the discussion above, we conclude that the number of keys assigned to a node is defined by the secure communication requirements and the topology of a system. The number of key distribution messages processed by a node is defined by the secure communication requirements, the topology, and the key distribution algorithm. The STB-D performs better when the key information can be piggybacked together for more nodes. For example, STB1 performs 150% better than STB2 in the Line, 130% better in the Grid, and 90% better in the Random deployment topologies.

B. Analytical Comparison

In this subsection, we compare our algorithm with the *Eschenauer-Gligor Scheme* and the *Du-Deng Scheme*. We focus on the number of keys a node has to store, the connectivity, and the resistance to node captures. Connectivity is defined as the probability that any two neighboring nodes share one key. Resilience is defined as a certain number of nodes that are captured by the adversaries, which compromise a certain fraction of the secure links.

Our algorithm assigns sufficient number of keys to each node based on the system topology and the secure communication requirements. Hence, the algorithm ensures that a shared key is provided for any two nodes that request communication. The number of keys per node is bounded by the number of its neighbors. A compromised node only reveals the keys it shares with its neighbors because no extra keys used by non-neighboring nodes are stored contrary to the pre-distribution algorithms.

As described in the *Eschenauer-Gligor Scheme*, if the number of nodes in the system is 10,000, the neighborhood size is 60 and when each node selects $\tau = 200$ keys from a key pool S, $|\mathrm{S}| = 100,000$, the probability that any two neighboring nodes share at least one key is 0.33. The exposure of one key leads to the compromise of another link with the probability of 0.3. As indicated in the *Du-Deng Scheme*, Equation 1 shows the relationship between the memory usage m and the

TABLE I
COMPARISONS OF THE THREE ALGORITHMS

	m = 60		connectivity = 0.98%	
	connectivity	resilient	m	resilient
Our Algorithm	100%	n-1	60	n-1
Esch-G Scheme	2%	1	-	-
Du-Deng Scheme	69%	19	80	19

number of τ spaces each node carries. m is defined in units of the key size, which is also the number of keys in our algorithm. λ represents the resilience degree to node captures. It means that as long as no more than λ nodes are compromised all the communication links of noncompromised nodes remain secure.

$$\tau = \lfloor \frac{m}{\lambda + 1} \rfloor \quad (1)$$

To ensure the fairness of algorithm comparison, we use the same network size (n) of 10,000 and neighborhood size of 60. We set the $\lambda = 19$ in Equation 1. The *Du-Deng Scheme* algorithm in Table I is the DDHV-D as described in [8].

As shown in Table I, the *Du-Deng Scheme* increases the local connectivity and resilience to node captures in comparison to the *Eschenauer-Gligor Scheme*. To achieve 98% connectivity the *Eschenauer-Gligor Scheme* requires that each node stores a much larger number of keys than in our approach. However, our algorithm can provide 100% connectivity with fewer keys and better resilience to node captures.

VIII. CONCLUSIONS

In this paper, we proposed a novel key distribution mechanism based on a precise post-deployment knowledge for wireless sensor networks. Our algorithm is composed of four phases – topology discovery, topology report, topology construction, and key distribution. It allows to assign keys based on the secure communication requirements of the system. A sufficient number of keys is assigned to each node. We studied two key assignment and two key distribution algorithms and the performance of their combinations. Based on the simulation results our approach is efficient in terms of memory, bandwidth and energy because the keys are assigned based on the system topology and communication requirements. When compared with *Eschenauer-Gligor Scheme* and *Du-Deng Scheme* our approach shows better performance in terms of connectivity and resilience to node captures.

REFERENCES

[1] R. Blom. An optimal class of symmetric key generation systems. In *Proceedings of EUROCRYPT84, Lecture Notes in Computer Science*, 1985.

[2] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conference. In *In Advances in Cryptology (CRYPTO'92)*, 1993.

[3] D.W. Carman, B.J. Matt, and G.H. Cirincione. Energy-efficient and low-latency key management for sensor networks. In *In Proceedings of 23rd Army Science Conference*, 2002.

[4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, 2003.

[5] W. Diffie and M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, 1976.

[6] W. Du, J. Deng, Y. S. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key pre-distribution scheme for wireless sensor networks. In *The ACM Transactions on Information and System Security (TISSEC)*, 2005.

[7] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, 2003.

[8] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A key predistribution scheme for sensor networks using depolyment knowledge. In *IEEE Transactions on Dependable and Secure Computing*, January-Mary 2006.

[9] W. Du, R. Wang, and P. Ning. An efficient scheme for authenticating public keys in sensor networks. In *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2005.

[10] L. Eschenauer and V. D. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security 2002*, 2002.

[11] G. Gaubatz, J.-P. Kaps, E. Ozturk, and B. Sunar. State of the art in public-key cryptography for wireless sensor networks. In *Second IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005)*, 2005.

[12] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, 2004.

[13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *In 10th ACM Conference on Computer and Communications Security*, 2003.

[14] D. Liu and P.Ning. Location-based pairwise key establishments for relatively static sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*, 2003.

[15] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *First IEEE International Conference on Sensor and Ad Hoc Communications and Networks SECON04*, 2004.

[16] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. In *IEEE Communications, vol. 32(9)*, 1994.

[17] R. D. Pietro, L. V. Mancini, and A. Mei. Random key assignment for secure wireless sensor networks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, 2003.

[18] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public key cryptosystems. In *Communications of the ACM*, 1978.